**TIMA**
Techniques de l'Informatique et de la Microélectroniques pour l'Architecture des systèmes intégrés
**Lab-STICC**
Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance

# Manuel d'Utilisation

**Version 1.0**
**2013**

Ahcène Bounceur

RMSCAT Platform

# RMS CAT Platform User Guide

## Product Version 1.0
## April 2013

© 2003-2013 RMSCATPlatform, Inc. All rights reserved.
Printed in France.

RMSCATPlatform, Inc., 46, Avenue Félix Viallet, 38031, Grenoble, France

**Trademarks:** Trademarks and service marks of RMSCATPlatform, Inc. (RMSCATPlatform) contained in this document are attributed to RMSCATPlatform with the appropriate symbol. For queries regarding RMSCATPlatform's trademarks, contact the corporate legal department at the address shown above or call +332-98-01-62-17.

All other trademarks are the property of their respective holders.

**Restricted Print Permission:** This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from RMSCATPlatform. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used solely for personal, informational, and noncommercial purposes;
2. The publication may not be modified in any way;
3. Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and
4. RMSCATPlatform reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from RMSCATPlatform.

**Disclaimer:** Information in this publication is subject to change without notice and does not represent a commitment on the part of RMSCATPlatform. The information contained herein is the proprietary and confidential information of RMSCATPlatform or its licensors, and is supplied subject to, and may be used only by RMSCATPlatform's customer in accordance with, a written agreement between RMSCATPlatform and its customer. Except as may be explicitly set forth in such agreement, RMSCATPlatform does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. RMSCATPlatform does not warrant that use of such information will not infringe any third party rights, nor does RMSCATPlatform assume any liability for damages or costs of any kind that may result from use of such information.

# Prise en main

## 1. Introduction

Ce manuel représente le guide de l'utilisateur de la plateforme RMSCATPlatform développée au sein du Laboratoire TIMA (groupe RMS). Cette plateforme est destinée pour l'évaluation d'un test analogique, mixte, Radio Fréquence et des MEMS (Micro Electro Mecanical Systems).

La plateforme RMSCATPlatform représente un ensemble d'outils de CAT (Computer Aided Test) incluant l'outil de simulation de faute Fidesim, l'outil de génération de vecteurs de test analogiques Optegen et l'outil d'évaluation de test Opteval. L'architecture générale représentant les différentes liaisons entre ces outils est donnée par la Figure 1.

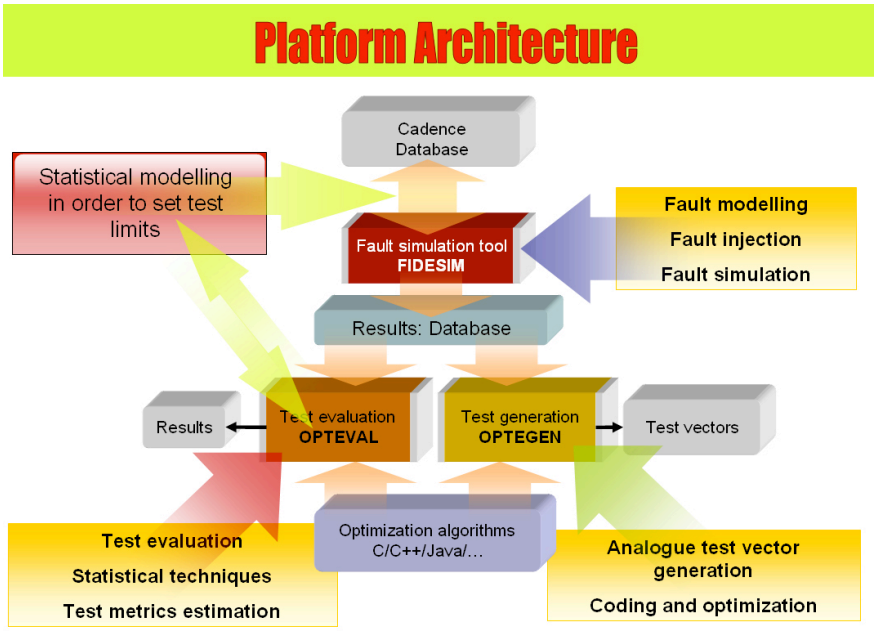



**Figure 1.1**

## 2. Circuit exemple

Pour illustrer dans ce manuel l'utilisation de la plateforme de test, un circuit utilisant un inverseur (Figure 1) sera utilisé comme exemple. Ce circuit sera appelé inverter_test. D'abord, nous allons créer l'inverseur lui-même qu'on appellera inverter. Les étapes suivantes montrent comment créer un inverseur sous Cadence.

### *2.1. Création de l'inverseur*

1. Dans le menu Cadence, cliquez sur File puis sur new puis sur Library. Une fenêtre New Library s'ouvre. Mettez dans le champ Name example, puis cliquez sur le bouton OK. Dans la fenêtre qui s'ouvre (Load Technology File) cliquez sur le bouton Cancel si vous ne connaissez pas le chemin du fichier de technologie utilisée.

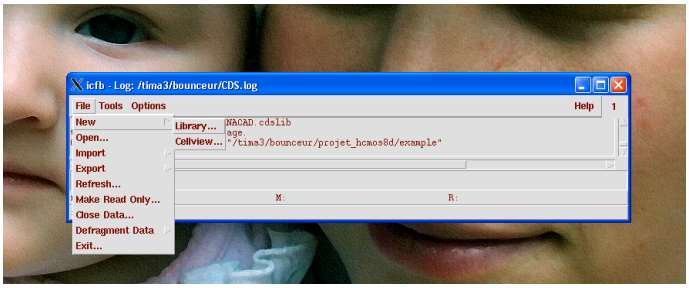
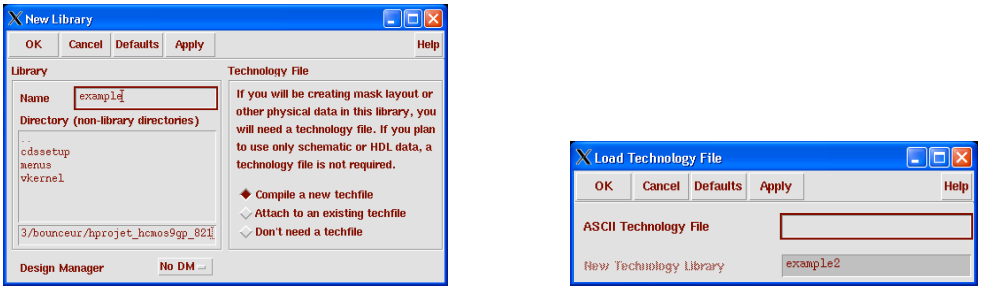



**Figure 2.1**




(a) (b)

**Figure 2.2**

2. Pour créer maintenant un inverseur qui fera partie de la libraire example, il faut créer une nouvelle cellule (forme schématique). Pour ce faire, dans le menu Cadence cliquez sur File puis sur New puis sur Cellview. La fenêter Create New File s'ouvrira. Mettez dans le champ Cell Name le mot inverter, dans View Name le mot shcematic et dans Tool choisissez Composer-Schematic.

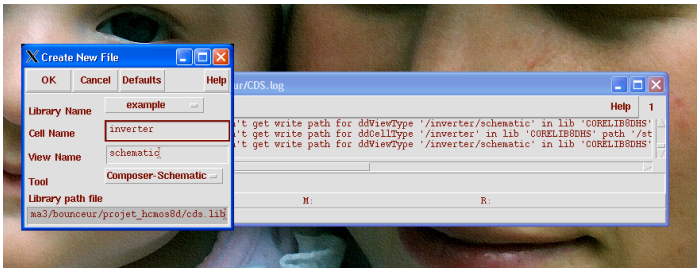



**Figure 2.3**

3. Créer un inverseur comme le montre la figure ci-dessous. Nommer le premier transistor en haut M1, le deuxième en bas M2, l'entrée Vin et la sortie Vout.

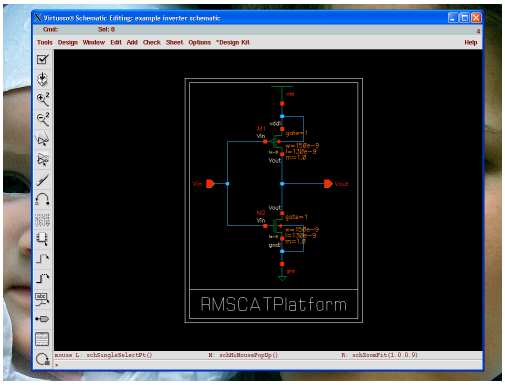



**Figure 2.4**

Maintenant l'inverseur est créé. Il reste que de lui créer un symbole, ce qui est le but de la section suivante.

## 2.2. Création d'un symbole inverseur

1. Ouvrez la forme schématique de l'inverseur créé. Dans le menu Design, choisissez Create Cellview puis From Cellview.

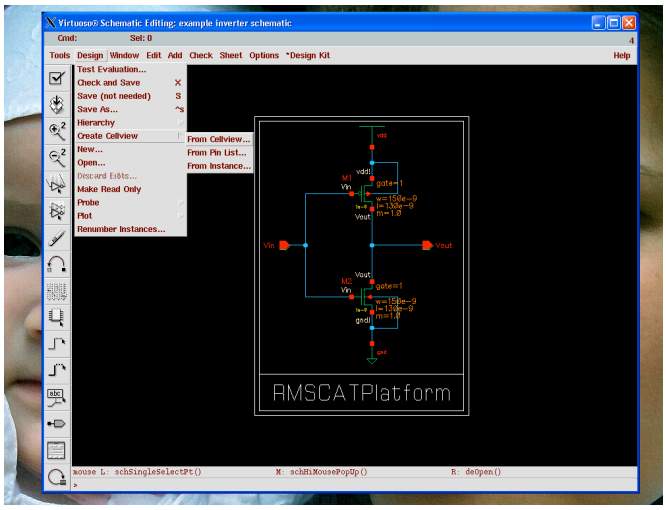



**Figure 2.5**

2. La finêtre Cellview From Cellview s'ouvre. Dans le champ Library Name mettez example, dans Cell Name mettez inverter puis cliquez sur le bouton OK.

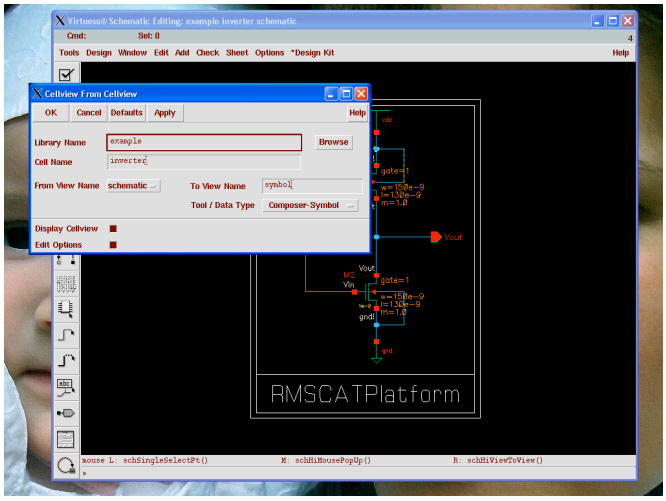



**Figure 2.6**

3. La fenêtre Symbol Generation Options s'ouvre. Cliquez directement sur le bouton OK.

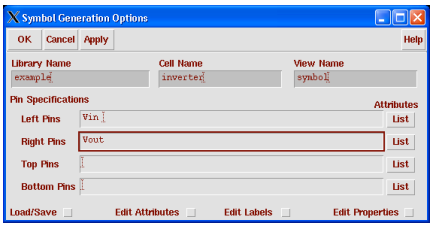
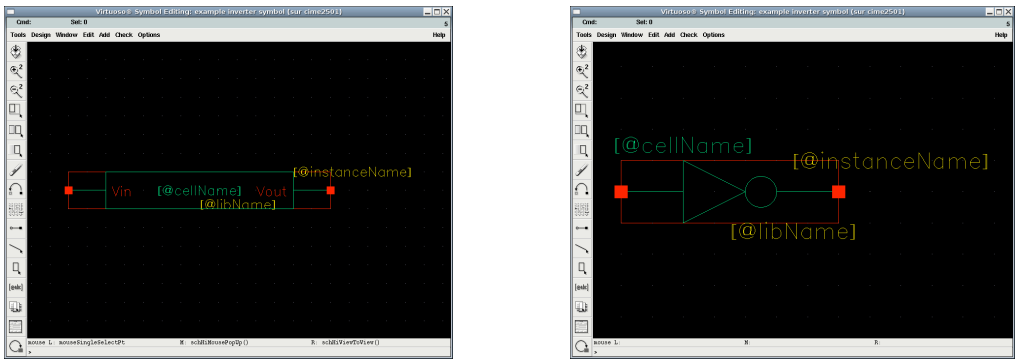



**Figure 2.7**

4. Un symbole par défaut sera créé, modifiez le pour qu'il ait la forme d'un inverseur comme le montre la Figure 9(b).




(a) (b)

**Figure 2.8**

## 2.3. *Création d'un circuit utilisant un inverseur*

La Figure 10(a) montre un exemple d'un circuit utilisant un inverseur appelé inverter_test. Il s'agit d'un circuit générant des pulsations (en utilisant un vpulse) pour vérifier ensuite si celles-ci sont inversées à la sortie de l'inverseur. Il est à noter que le nom donné à l'inverseur est I2, le nom donné au net sortant de vpulse V1 est net1 et le nom donné au net sortant de l'inverseur I2 est net2. Les différentes configurations du vdc et du vpulse sont données par les Figures 10(b) et 10(c).

(a) (b) (c)

**Figure 2.9**

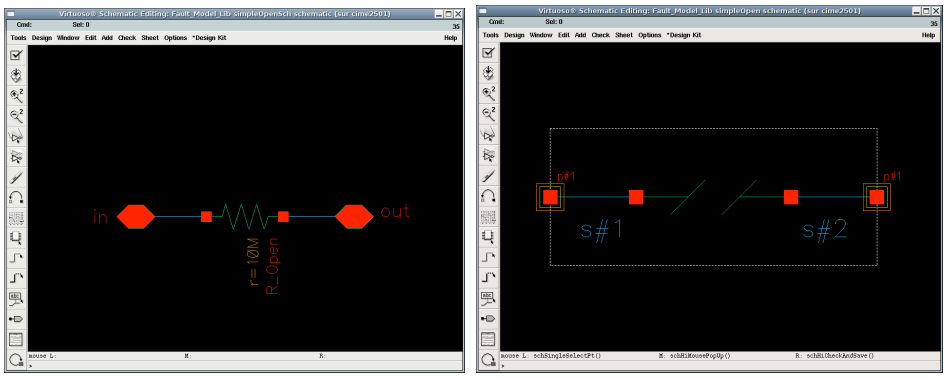
La simulation de ce circuit sous l'analyse transitoire à 300m devrait donner le résultat de la Figure 11.

**Figure 2.10**

# 3. Simulation de fautes

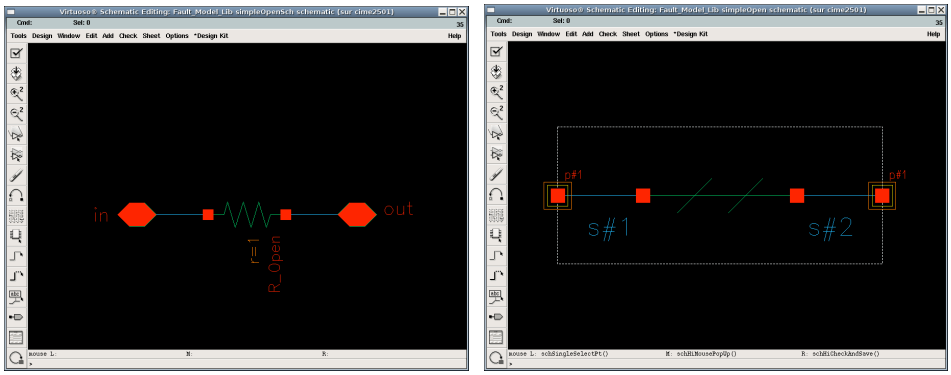
## *3.1. Création des modèles de fautes catastrophiques*

Les modèles de fautes sont créés comme tout circuit Cadence. Quelques modèles les plus classiques, tels que le circuit-ouvert (open) et le court-circuit (short) sont déjà fournit avec la plateforme. Ceux-ci, ce trouvent dans la librairie Fault_Model_Lib sous les noms de cellules simpleOpen (Figure 12(a) la forme schématique, Figure 12(b) le symbole) et simpleShort (Figure 13(a) la forme schématique, Figure 13(b) le symbole), repectivement.




(a) (b)

**Figure 3.1**




(a) (b)

**Figure 3.2**

## 3.2. Création des modèles de fautes paramétriques

Les fautes paramétriques sont de deux types : explicites et implicites. Les fautes paramétriques explicites s'injectent comme les fautes catastrophiques. C'est-à-dire que les déviations sur les paramètres sont choisies à priori. En revanche, les fautes paramétriques implicites ne sont pas connue à priori, car il s'agit de pour chaque paramètre de sa déviation minimale permettant de violer au moins une des spécifications du circuit sous test. La description de l'injections des fautes paramétriques peuvent être faite en utilisant le descripteur FIDesc (voir la section Décrire comment injecter des fautes).

## 3.3. Création des blocs fautifs

La création d'un bloc fautif n'est nécessaire que si la faute doit être injectée dans un composant qui se trouve dans un bloc ou sous bloc du circuit sous test. Car dans ce cas de figure, si une faute est injectée directement dans un bloc, et que dans le même circuit il existe un autre bloc du même type (même librairie et même cellule) elle sera aussi injectée dans les autres. Pour remédier à ce problème, il est donc nécessaire de créer une copie exacte de ce bloc qui remplacera donc le bloc contenant le composant où la faute sera injectée. Ainsi, la faute sera injectée uniquement dans ce bloc. La plateforme remplacera ce bloc automatiquement et le supprimera une fois la simulation de fautes terminée. L'utilisateur n'a qu'à créer une copie de ce bloc. Il est aussi recommander d'ajouter une croix ou un signe au symbole du bloc fautif montrant que ce bloc est le bloc fautif.

Dans le cas du circuit inverseur_test, on a bien à faire à ce cas de figure car certaines fautes doivent être injectées dans l'inverseur. Ce dernier représente un bloc (nom I2). Il est donc nécessaire de créer une copie exacte, de ce bloc, qui sera sauvegardée dans la libraire Fault_Model_Lib et qui aura le nom inverter_faulty. Pour ce faire, il faut ouvrir la manager des librairies (Library Manager). Ceci se fait à parti du menu outils de Cadence.

1. une fois le Library Manager est ouvert, sélectionnez la libraire example puis la cellule inverter, puis cliquez sur le bouton droit de la souris en le maintenant enfoncé. Un sous menu apparaîtra dans lequel il faut cliquez sur copy.

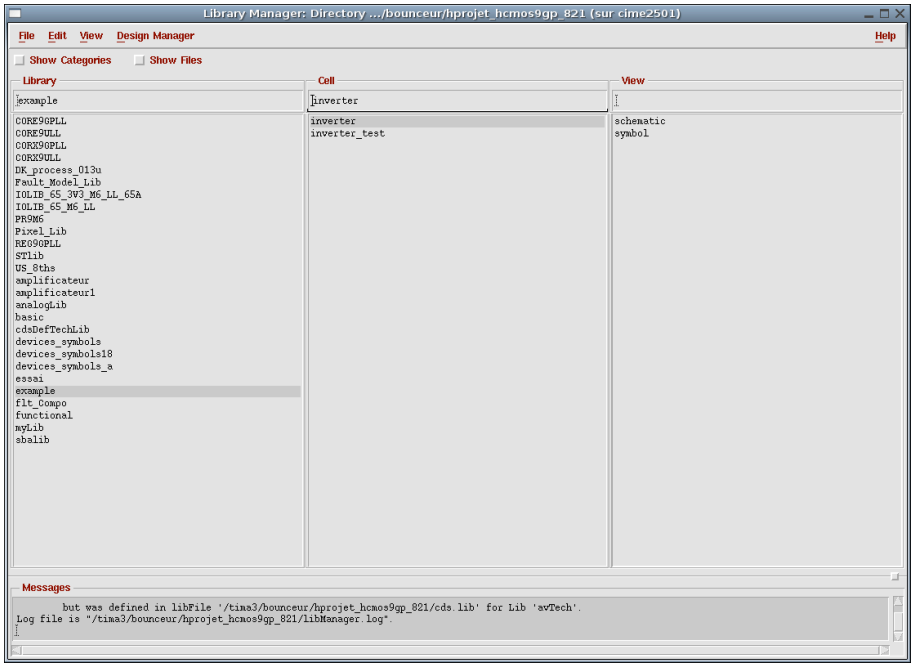



**Figure 3.3**

2. La fenêtre de la Figure 15 sera ouverte. Dans celle-ci, dans la partir To, Fault_Model_Lib dans le champ Library et inverter_faulty dans le champ Cell. Une autre fenêtre sera ouverte (Figure 16). Cliquez directement sur le bouton OK. Une copie exacte du bloc inverter est créée et se trouve dans la libraire Fault_Model_Lib.

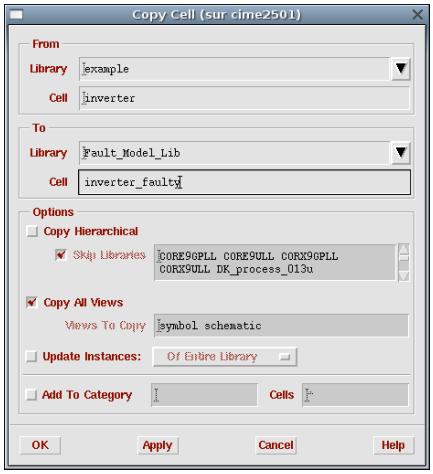



**Figure 3.4**

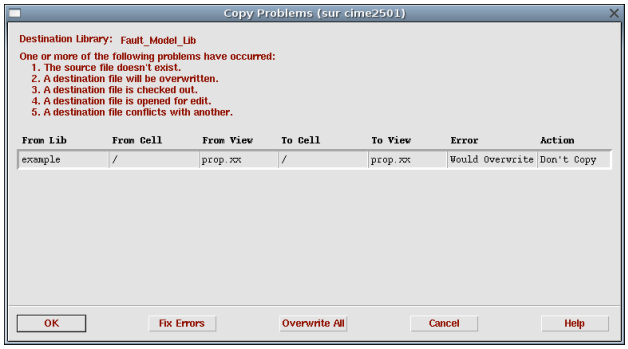

Copy Problems (sur cime2501)
Destination Library: Fault_Model_Lib
One or more of the following problems have occurred:
1. The source file doesn't exist.
2. A destination file will be overwritten.
3. A destination file is checked out.
4. A destination file is opened for edit.
5. A destination file conflicts with another.
From Lib From Cell From View To Cell To View Error Action
example / prop.xx / prop.xx Would Overwrite Don't Copy
OK Fix Errors Overwrite All Cancel Help

**Figure 3.5**

3. Maintenant pour différentier le symbole d'un inverseur sans fautes (inverter) de l'inverseur avec fautes (inverter_faulty) il faut juste changer la forme de son symbole. Pour se faire, il faut ouvrir à partir de Library Manager, le symbole de inverter_faulty. Donc, cliquez sur la librairie Fault_Model_Lib, puis dans la cellule inverter_faulty, puis cliquez deux fois sur le mot symbol qui se trouve dans la partie View.

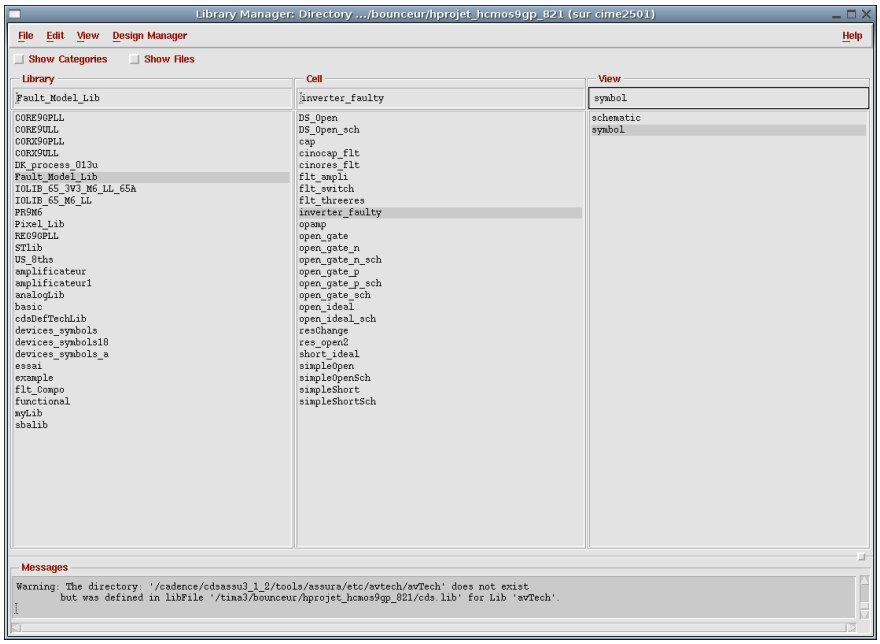



**Figure 3.6**

4. Modifiez donc ce symbole sans surtout déplacer ses ports (pins). Il est recommandé d'ajouter une croix comme le montre la Figure 18.

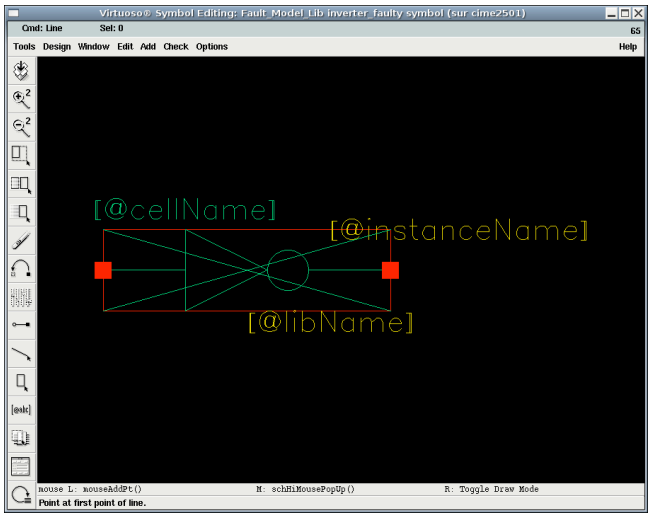

Virtuoso® Symbol Editing: Fault_Model_Lib inverter_faulty symbol (sur cime2501)
Cmd: Line Sel: 0
Tools Design Window Edit Add Check Options Help
[@cellName]
[@instanceName]
[@libName]
mouse L: mouseAddPt() M: schHiMousePopUp() R: Toggle Draw Mode
Point at first point of line.

**Figure 3.7**

## *3.4. Décrire comment injecter des fautes*

### 4.3.1. Le Descripteur des Injections de Fautes FIDesc

Une fois tous les modèles de fautes sont crées ainsi que tous les blocs fautifs, il est donc nécessaire de créer les différents fichiers (appelés FID[1]). Pour ce faire, il faut

[1] Fault Injection Description.

utiliser le descripteur des injections de fautes (Fault Injection Descriptor) appelé FIDesc montré par la Figure 19.

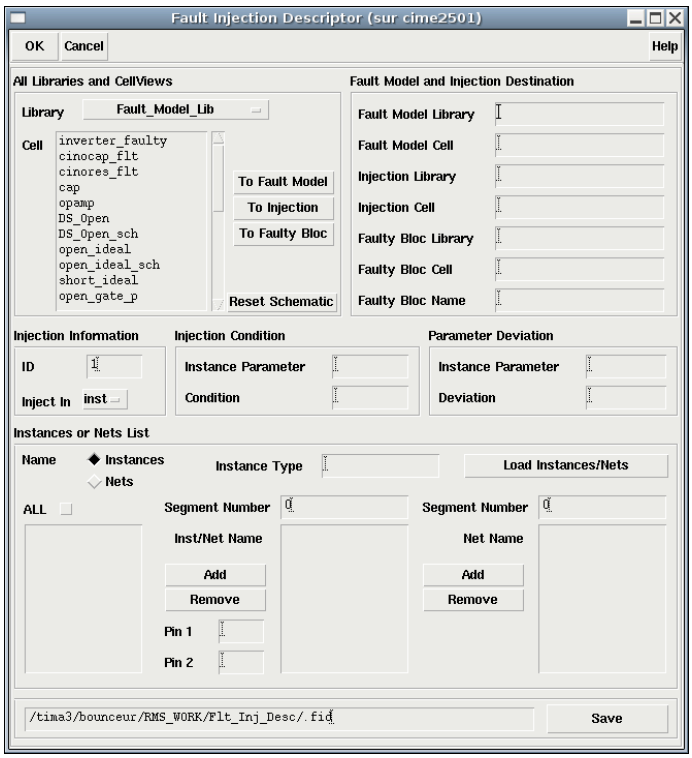



**Figure 3.8**

### 4.3.2. Décrire des injections de fautes catastrophiques

Comme exemple pour le cas du circuit inverter_test, nous allons considérer l'injection des courts-circuits entre les différents ports de chacun des deux transistors (M1 et M2) de l'inverseur et entre les deux ports de la résistance R0. On ajoutera à ceci deux circuits-ouverts au niveau de la grille de chacun des deux transistors. Ce qui résulte en 9 injections de fautes.

Pour créer le fichier FID permettant d'injecter un circuit-ouvert à la grille du transistor M1, il faut suivre les étapes suivantes :

1. Choisissez le modèle de faute représentant le circuit de la libraire Fault_Model_Lib (dans la partie Library) et de la cellule simpleOpen (dans la partir Cell) puis cliquez sur le bouton To Fault Model.

2. Choisissez le circuit où cette faute sera injectée. Dans, notre cas, ce circuit représente le circuit inverter_faulty, car en réalité la faute ne sera pas injectée directement dans le bloc inverter I2, mais dans le bloc inverter_faulty qui remplacera automatiquement le bloc inverter I2 tout en gardant le nom I2. Donc, dans FIDesc, choisissez la libraire Fault_Model_Lib puis la cellule inverter_Faulty, puis cliques sur le bouton To Injection.

3. Désignez le nom du bloc qui sera par le bloc fautif. Dans notre cas, il y a un seul bloc qui est le bloc I2 (l'inverseur). Mettez donc dans le champ Faulty Block Name le nom I2. Dans les champs Faulty Block Library et Faulty Block Cell, il faut mettre le nom de la libraire et le nom de la cellule du circuit contenant le bloc où la faute sera

injectée, respectivement. Ce circuit peut être le circuit sous test comme il peut représenter un sous bloc d'un des blocs du circuit sous test. Dans notre cas, ce bloc se trouve directement dans le circuit sous test (le circuit inverter_test). Donc, il faut mettre dans le champ Faulty Block Library le nom example et dans le champ Faulty Block Cell le nom inverter_test.

4. Maintenant, clique sur le bouton Load Instances/Nets. Tous les noms des composants (instances) de la cellule Injection Cell seront listés.

5. Sélectionnez une des instances listées. Le type de cette instance sera affiché dans le champ Instance Type. Pour afficher uniquement les instances de ce type, cliquez à nouveau sur le bouton Load Instances/Nets. Si à nouveau vous voulez afficher la liste de toutes les instances, il suffit d'effacer le contenu la case Instance Type. Il est fortement conseillé d'afficher à chaque fois la liste des instances du même type.

6. A l'aide du bouton Add (à gauche) ajouter toutes les instances où la faute sera injectée, pour l'inverseur, nous allons ajouter M1. Le bouton Remove permet de supprimer une instance ajoutée par erreur. Dans le champ Pin1 mettez le numéro 1. C'est-à-dire le numéro du port de l'instance M1 représentant la grille. Ce numéro est donné par la fonction selectPin qui sera décrite dans la prochaine section.

7. Donner un nom à ce fichier décrivant l'injection d'un circuit ouvert au niveau de la grille du transistor M1. Donnez le nom open_G_M1.fid (Figure 20(a)).

8. A l'aide du bouton Remove enlever M1 et ajouter M2 puis Remplacer le numéro 1 par 2 dans le champ Pin1. Sauvegarder ensuite le fichier FID décrivant l'injection d'un circuit ouvert au niveau de la grille du transistor M2. Donnez le nom open_G_M2.fid (Figure 20(b)). Il est à noter que le numéro du port d'un transistor représentant la grille n'est pas le même pour les nmos et les pmos.

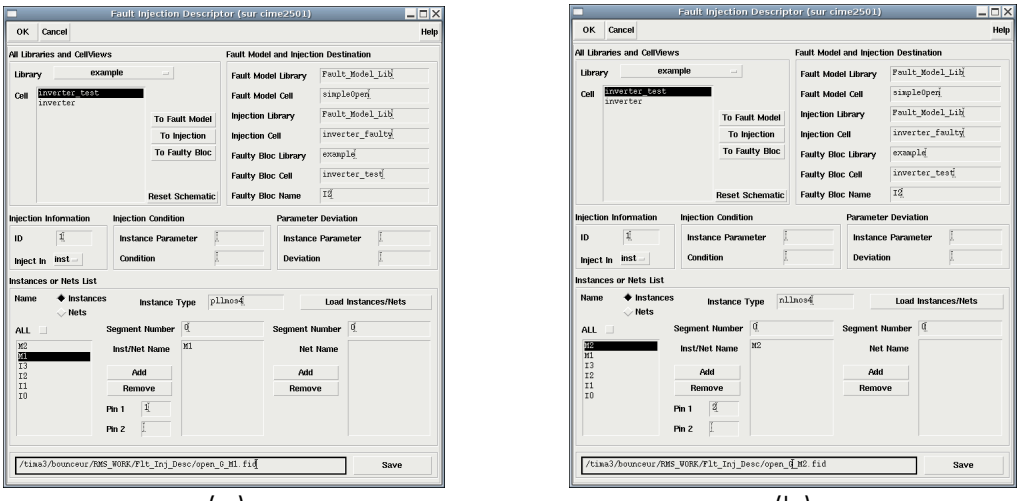



**Figure 3.9**

9. Pour créer un fichier FID permettant d'injecter un court circuit entre les deux pins de la résistance R0, il faut choisir la librairie example et la cellule inverter_test puis cliquez sur To Injection, car cette fois ci, la faute ne sera pas injectée dans un bloc mais directement dans la résistance du circuit sous test inverter_test. Les trois champs du Faulty Block (Library, Cell, Name) doivent être vides. Il est à noter que les champs

Injection Library et Injection Cell peuvent être vide si le circuit représente le circuit sous test (inverter_test dans notre cas).

10. Cliquez sur le bouton Load Instances/Nets et choisissez la résistance R0. Cliquez ensuite sur la bouton Add. Mettez 1 dans le champ Pin 1 et 2 dans le champ Pin 2. Sauvegardez le fichier sous le nom short_R0.fid (Figure 22).

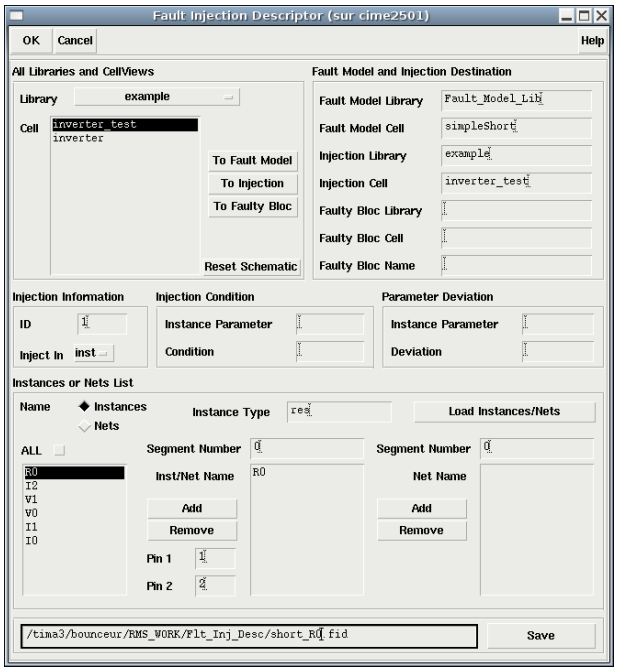



**Figure 3.10**

11. Créez le reste des fichiers FID. Et donnez leurs les noms comme suit :
    - Circuit ouvert entre la grille et la source du transistor M1 : 1-short_GS_M1.fid
    - Circuit ouvert entre la grille et le drain du transistor M1 : 2-short_GD_M1.fid
    - Circuit ouvert entre la source est le drain du transistor M1 : 3-short_SD_M1.fid
    - Circuit ouvert entre la grille et la source du transistor M2 : 4-short_GS_M2.fid
    - Circuit ouvert entre la grille et le drain du transistor M2 : 5-short_GD_M2.fid
    - Circuit ouvert entre la source est le drain du transistor M2 : 6-short_SD_M2.fid
    - Court circuit au niveau de la grille du transistor M1 : 7-open_G_M1.fid
    - Court circuit au niveau de la grille du transistor M2 : 8-short_G_M2.fid
    - Circuit ouvert entre les pins de la résistance R0 : 9-short_R0.fid

### 4.3.3. Décrire des injections de fautes paramétriques

Les fautes paramétriques explicites sont créées en utilisant le descripteur FIDesc. Les fautes paramétriques implicites sont décrites en créant manuellement un fichier FID spécifique.

La Figure 23 montre un exemple d'un fichier FID permettant d'injecter une déviation de 50% sur le paramètre r (résistance) de la résistance R0. Pour cela, il suffit de mettre le nom du circuit contenant la résistance R0 (pour le cas du circuit inverter_test, il faut mettre example dans Injection Librairie et inverter_test dans Injection Cell). Mettez ensuite r dans Instance Parameter et 50 dans Deviation. Donnez ensuite le nom dev_r_50.fid au fichier FID à créer. Cet exemple montre comment décrire l'injection d'une faute paramétrique explicite.

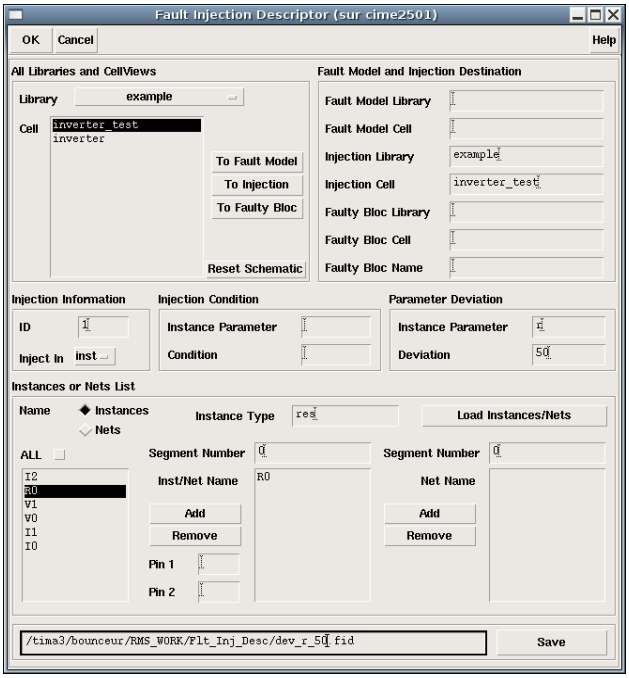



**Figure 3.11**

L'injection de cette faute est montrée pas la Figure 24.

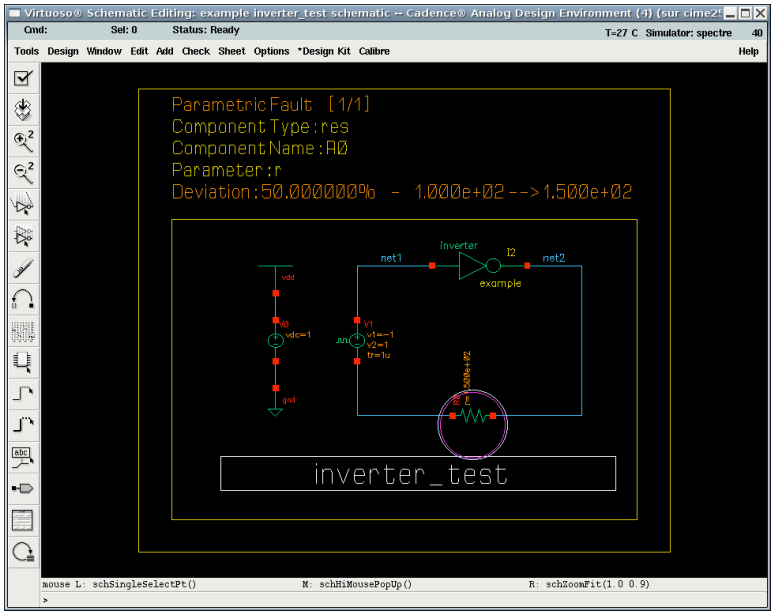



**Figure 3.12**

Pour créer une faute paramétrique implicite il faut créer un fichier FID manuellement. Ce fichier contient plusieurs ligne de la forme suivante :

```
(("M1" "M2") "type" "w" "Fault_Model_Lib" "Faulty_circuit" "Block1" 3u 100)
```

où,

**("M1" "M2")** : eprésente la liste des noms des instances la faute sera injectée. Dans ce cas, il s'agit de transistors.
**"type"** : représente le type des instances M1 et M2. Par exemple, npn ou pnp.

**"w"** : représente le paramètre à varier.
**"Fault_Model_Lib"** : représente la librairie de la cellure Faulty_circuit. Cette partie peut être remplacée par une chaine vide **""** si les transistors se trouvent directement dans le circuit sous test.
**"Faulty_circuit"** : représente la cellule contenant le bloc Block1. Cette partie peut être remplacée par une chaine vide **""** si les transistors se trouvent directement dans le circuit sous test.
**"Block1"** : représente le bloc contenant M1 et M2 où la faute sera injectée. Cette partie peut être remplacée par une chaine vide **""** si les transistors se trouvent directement dans le circuit sous test et pas dans ce bloc.
**3u** : représente la valeur typique du paramètre w. Il est à noter que le w de M1 et M2 doit avoir la même valeur typique, sinon, il faut créer deux lignes différente pour chaque paramètre.
**100** : une valeur en pourcentage qui désigne la déviation maximale sur le paramètre w.

Une fois ce fichier est créé, il reste de lancer les injections décrites dans celui-ci. Ceci se fait en choisissant Parametric Fault Simulation qui se trouve dans le menu Tools. La fenêtre de la Figure 25 sera ouverte. Il suffit juste de choisir le nombre d'itérations (dans Iteration Number) nécessaire pour trouver la valeur minimale de chaque paramètre permettant de violer au moins une des spécifications du circuit sous test. Puis, cliquez sur le bouton Search Parameter Limits.

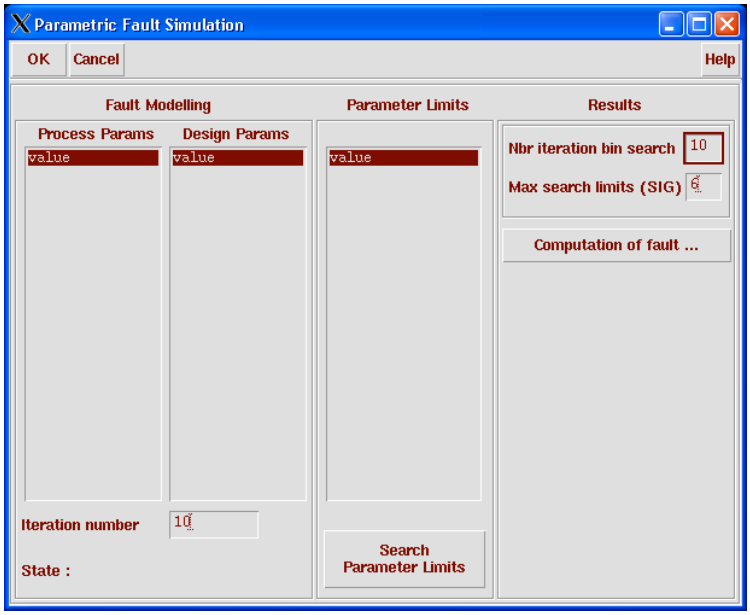



**Figure 3.13**

### 4.3.4. La fonction selectPin

Pour connaître le numéro d'un port d'une instance, il faut utiliser la fonction selectPin qui permet de sélectionner à l'aide d'un cercle le port de l'instance donné en paramètre. Cette fonction possède trois paramètres. Le premier paramètre représente le numéro de la fenêtre du circuit (schématique) où se trouve l'instance. Le deuxième paramètre représente le nom de l'instance. Le troisième paramètre représente le numéro du port à sélectionner de cette instance. La Figure 23 montre le circuit inverter ouvert dans une fenêtre schématique ayant le numéro 7. Nous

avons utilisé la commande selectPin pour sélectionner le port numéro 2 du transistor M1. Cette commande est utilisée dans la fenêtre Cadence comme suit :

```
selectPin(7 "M1" 2)
```

Un cercle orange (Figure 23) permet de voir quel port du transistor M1 ayant le numéro 2.

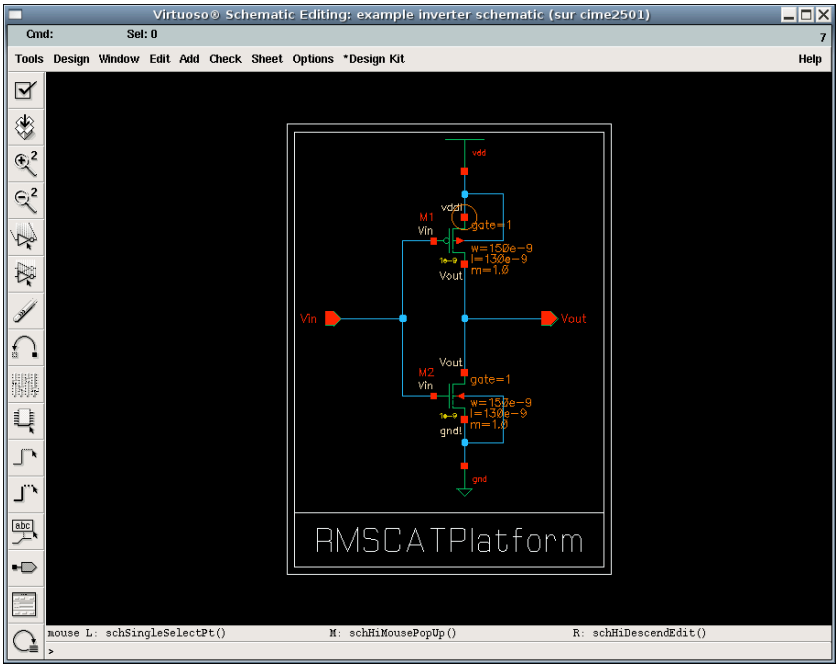



**Figure 3.14**

## 3.5. *La simulation de fautes catastrophiques*

1. Pour effectuer une simulation de fautes, il faut en premier lancer l'outil OPETVAL en utilisant le menu Cadence → Tools → RMS Tools → Test Evaluation (Figure 24).

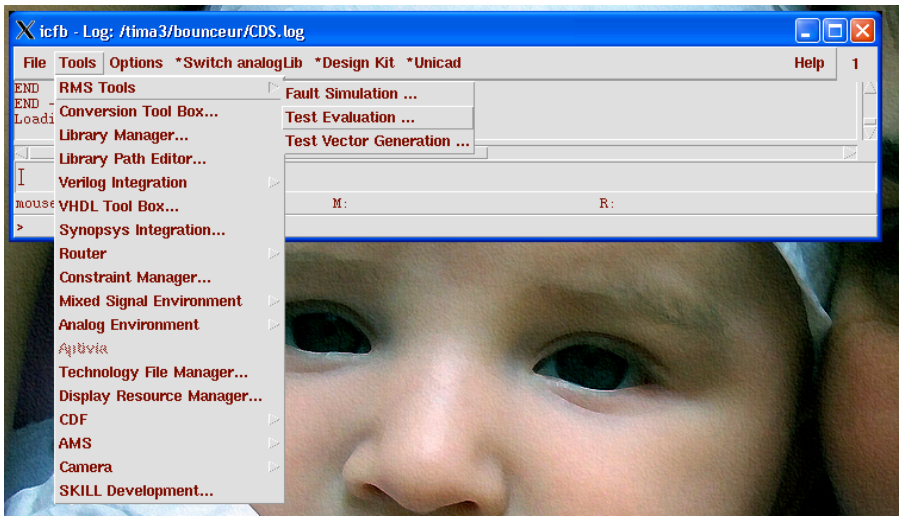



**Figure 3.15**

2. L'outil de l'évaluation de test OPTEVAL sera ouvert (Figure 25).

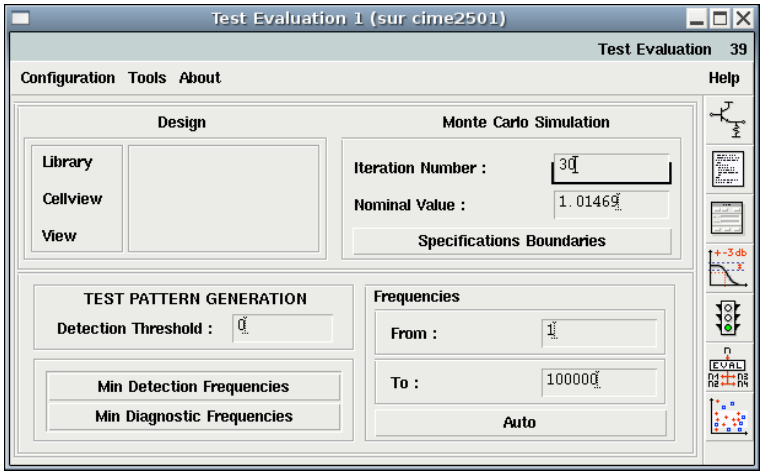



**Figure 3.16**

3. Cliquez sur menu Configuration → Set Design, puis choisissez le circuit inverter_test qui se trouve dans la librairie example (Figure 26). Ce circuit représente le circuit sous test.

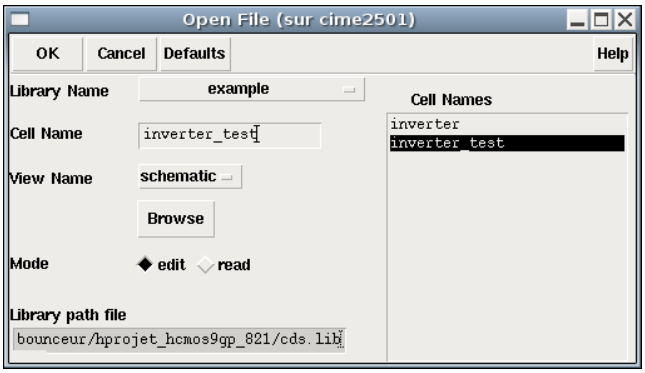



**Figure 3.17**

La fenêtre analog environment sera automatiquement ouverte. Il faut la configurer en entrant toutes les informations nécessaire pour la simulation. Pour notre exemple nous avons choisi une simulation transitoire à 100m.

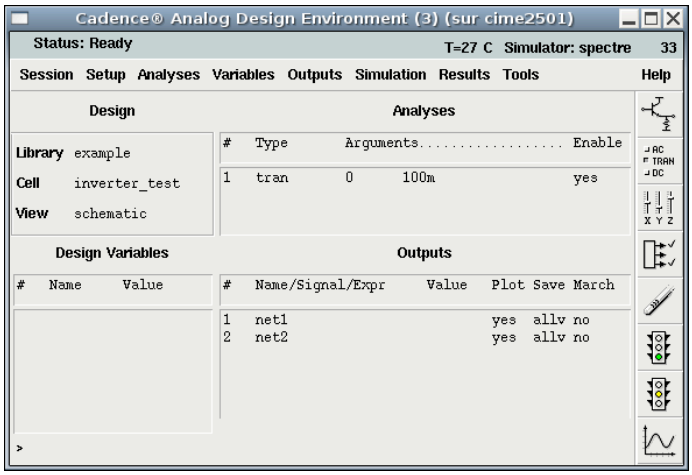



**Figure 3.18**

4. Dans le menu Configuration, cliquez sur Test Configuration Program. Puis entrer le programme de configuration de test de la Figure 28. Puis cliquez sur le bouton OK.

Dans les prochaines sections ce programme sera décrit en détail. C'est le programme où seront décrites les différentes performances et leurs spécifications ainsi que les différents critères de test et leurs limites de test.

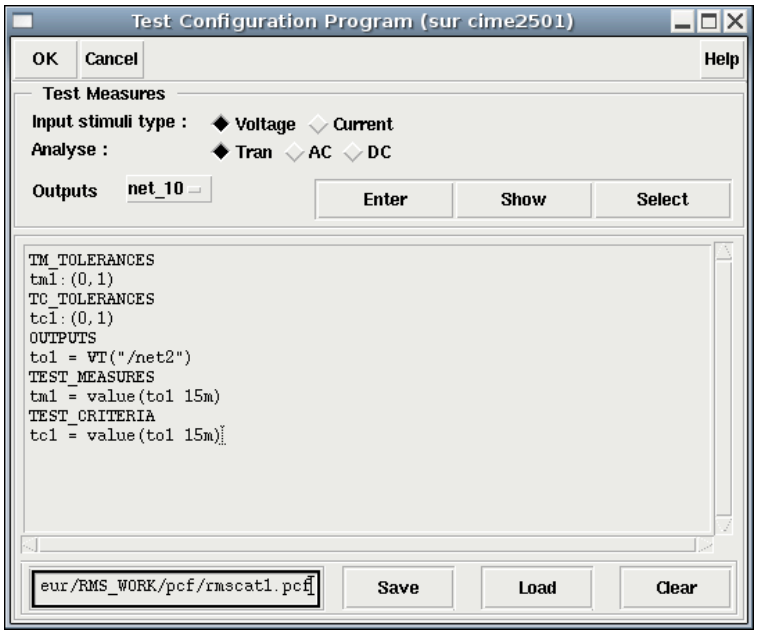



**Figure 3.19**

5. Dans le menu Tools, cliquez sur Single Fault Simulation. L'outil de simulation de fautes FIDESIM sera ouvert (Figure 29). Il est à noter que la partie Faults est vide et elle sera disponible une fois les fautes sont injectées.

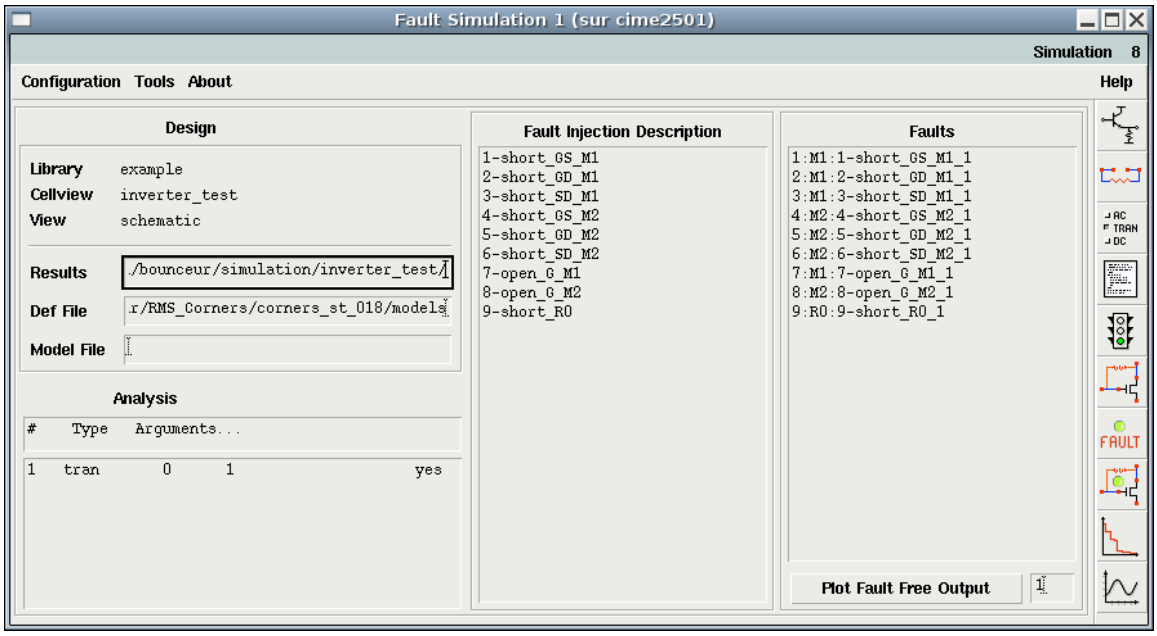



**Figure 3.20**

6. Dans le menu Configuration de l'outil FIDESIM, choisissez Fault Model Path. La fenêtre de la Figure 30 sera ouverte. Dans le champ Cata FID Path faites entrer le chemin où vous avez enregistré les fichiers FID créés précédemment.

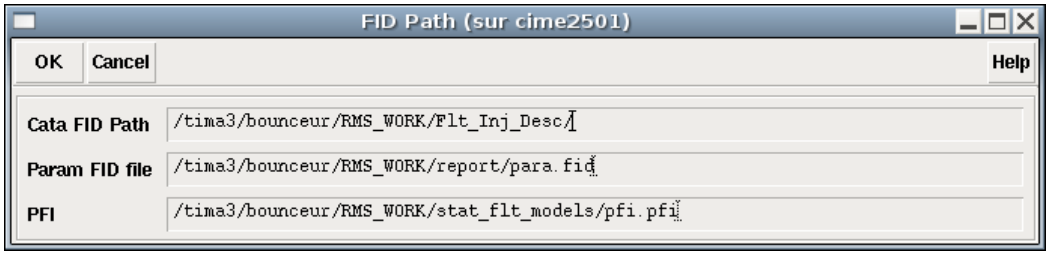



**Figure 3.21**

7. Toujours dans le menu Configuration, choisissez Choose Fault Model. La fenêtre de la Figure 31 sera ouverte. La liste des FIDs créés figure dans la partie Complete Fault Model List. Sélectionner les 9 fichiers FID précédemment créés puis cliquez sur le bouton Add.

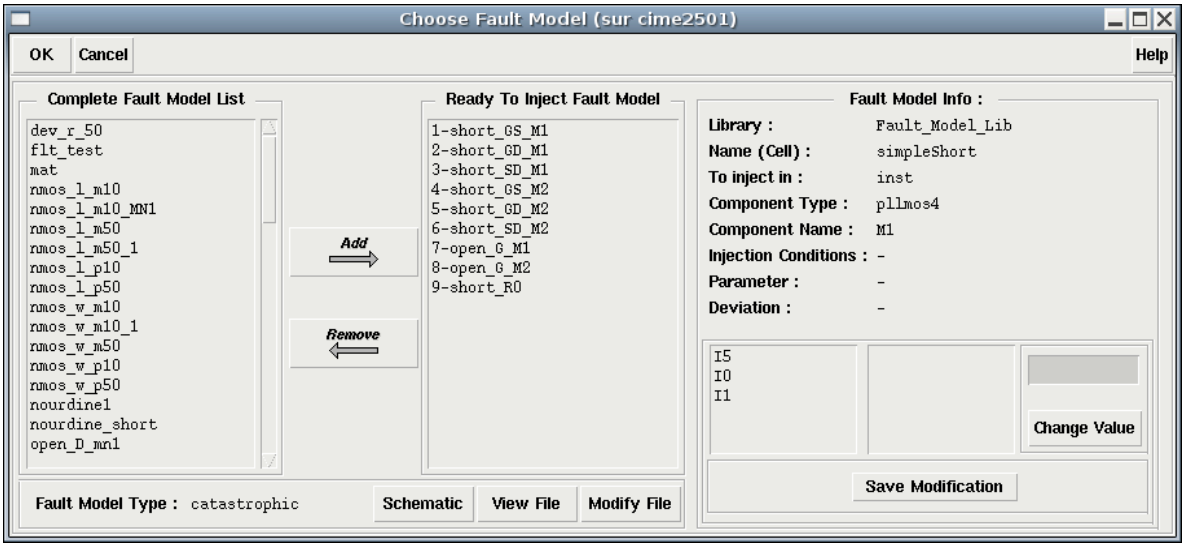



**Figure 3.22**

Cette fenêtre permet d'ouvrir un fichier FID sélectionné pour le voir (bouton View File) et pour le modifier (bouton Modify File).
8. Dans le menu Configuration, choisissez maintenant Analysis et Variables. Puis cliquez directement sur le bouton OK. Ceci parce que l'outil récupère automatiquement la configuration faite dans analog environment. La version actuelle de l'outil permet de faire ça uniquement avec les simulations de type transitoire, fréquentiel. Sinon, avant de cliquer sur le bouton OK de la fenêtre Analysis and Variables, il faut cliquer sur Script Simulation (Figure 32) et ajoutez le script de simulation. Ce script peut être récupéré directement à partir de analog environment.

Il est à noter que si vous utilisez une certaine technologie, vous aurez besoin d'utiliser des fichiers modèles. Ces fichiers sont généralement contenus dans le fichier corners.scs qui se trouve dans le répertoire netlist du répertoire simulation. Souvent, il est sous la forme suivante :
/home/simulation/nom_cellule_circuit/spectre/schematic/netlist/corners.scs
Donc il faut remplacer dans le script de simulation donné par Cadence le mot models par ce chemin.

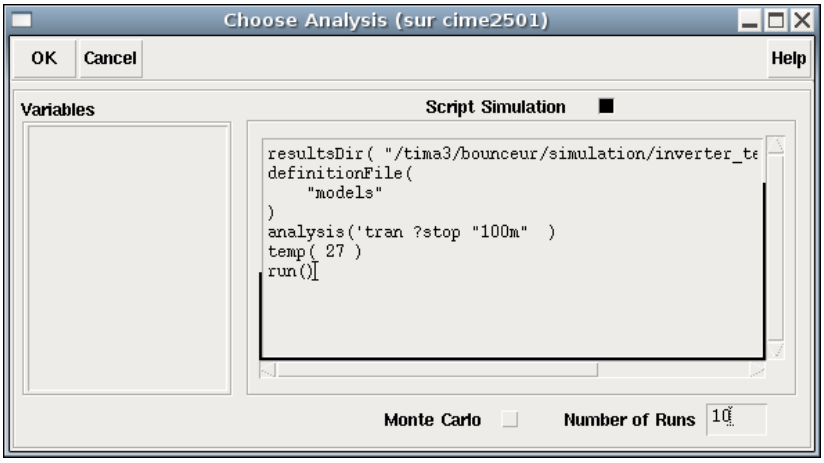



**Figure 3.23**

9. La récupération du script de simulation se fait comme suit. Dans le menu Session de analog environment, choisissez save script. Puis enregistrez le script sous le nom que vous voulez (Figure 33).

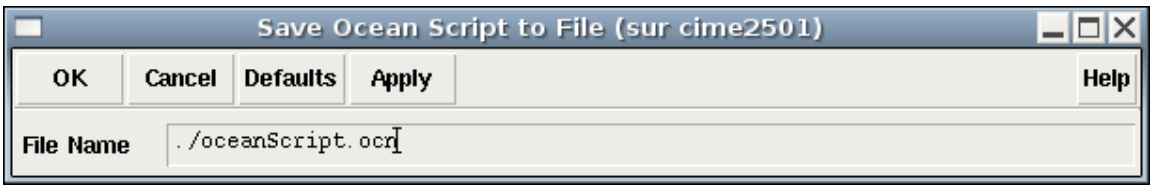



**Figure 3.24**

Un fichier script exemple est montré par la Figure 34. Sélection la partie se trouvant entre resultsDir et run(). Collez cette partir dans la partie script simulation de la Figure 32.

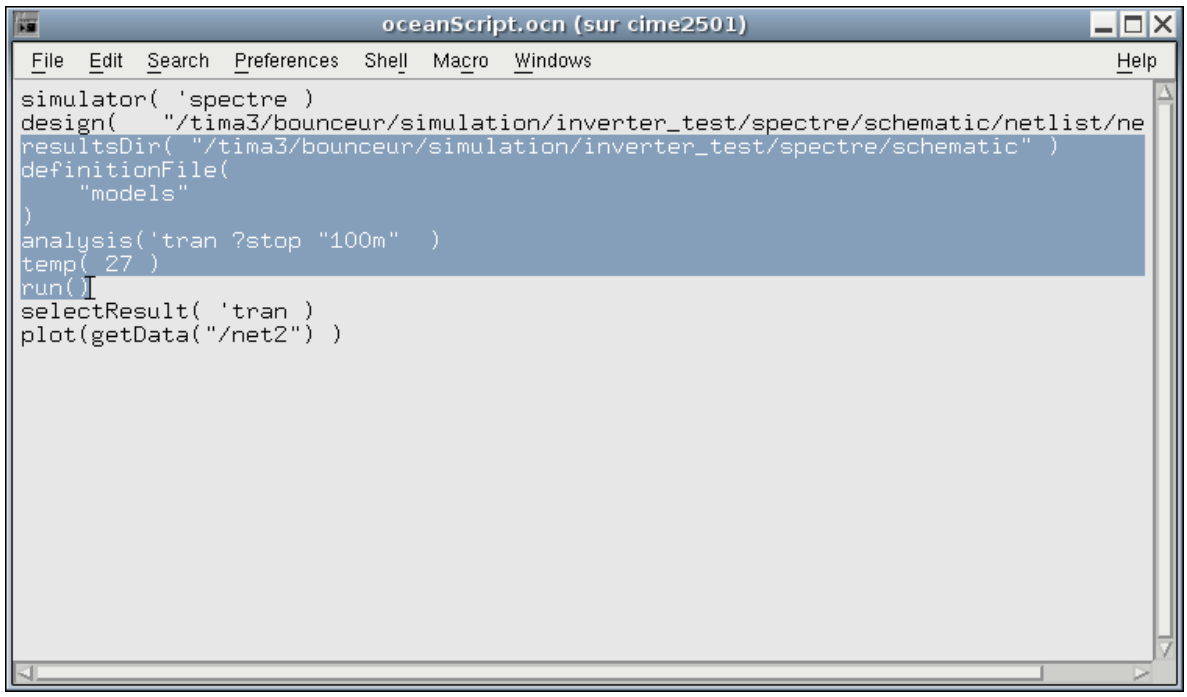



**Figure 3.25**

10. Dans le menu Configuration, choisissez Test Configuration, puis cliquez sur le bouton OK (Figure 35). Même si le programme n'est pas modifié, il est nécessaire d'ouvrir cette fenêtre et de cliquer sur le bouton OK.

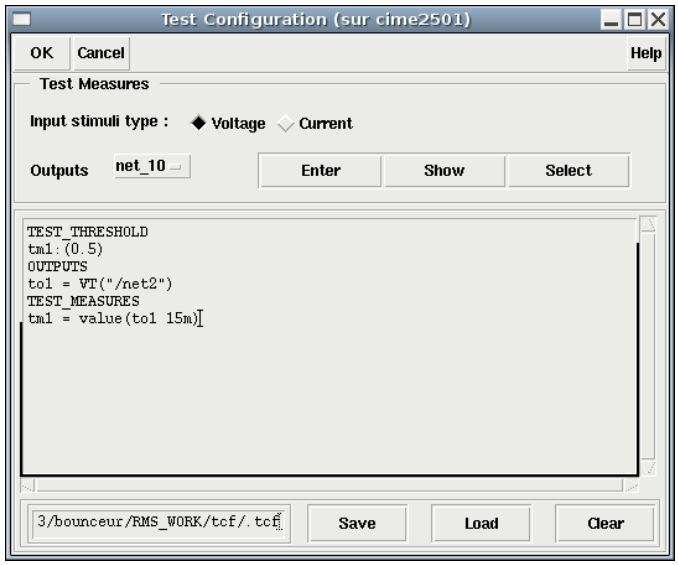



**Figure 3.26**

11. Dans le menu tools, cliquez sur Fault Free Simulation. Attendez jusqu'à ce que la simulation soit finie. Si aucune erreur ne s'est produite dans le programme de configuration, les graphes des différentes sorties (outputs) du programme de configuration seront affichés (comme le montre la Figure 36 pour le cas du circuit inverter_test).

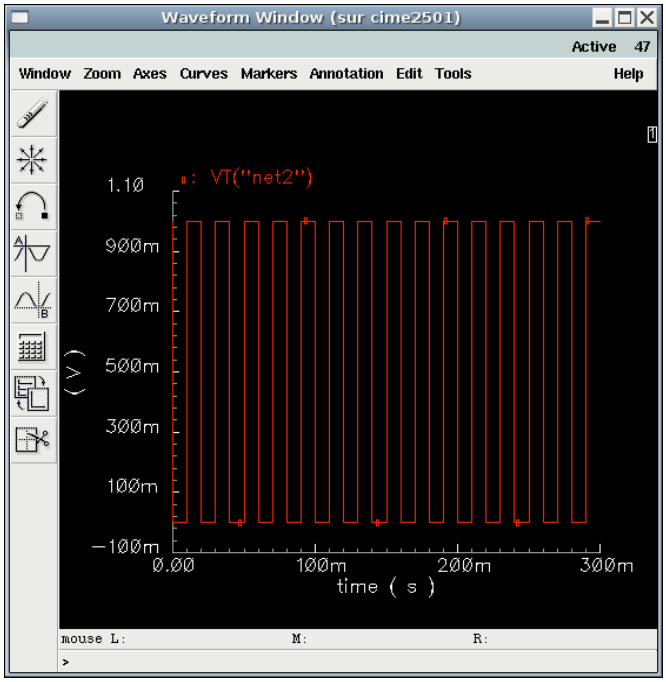



**Figure 3.27**

12. Ensuite, choisissez dans le menu tools, Fault simulation, inject and simulate. La compagne de l'injection de fautes sera lancée. Les figures 37(a) à (i) montrent les 9 fautes injectées.

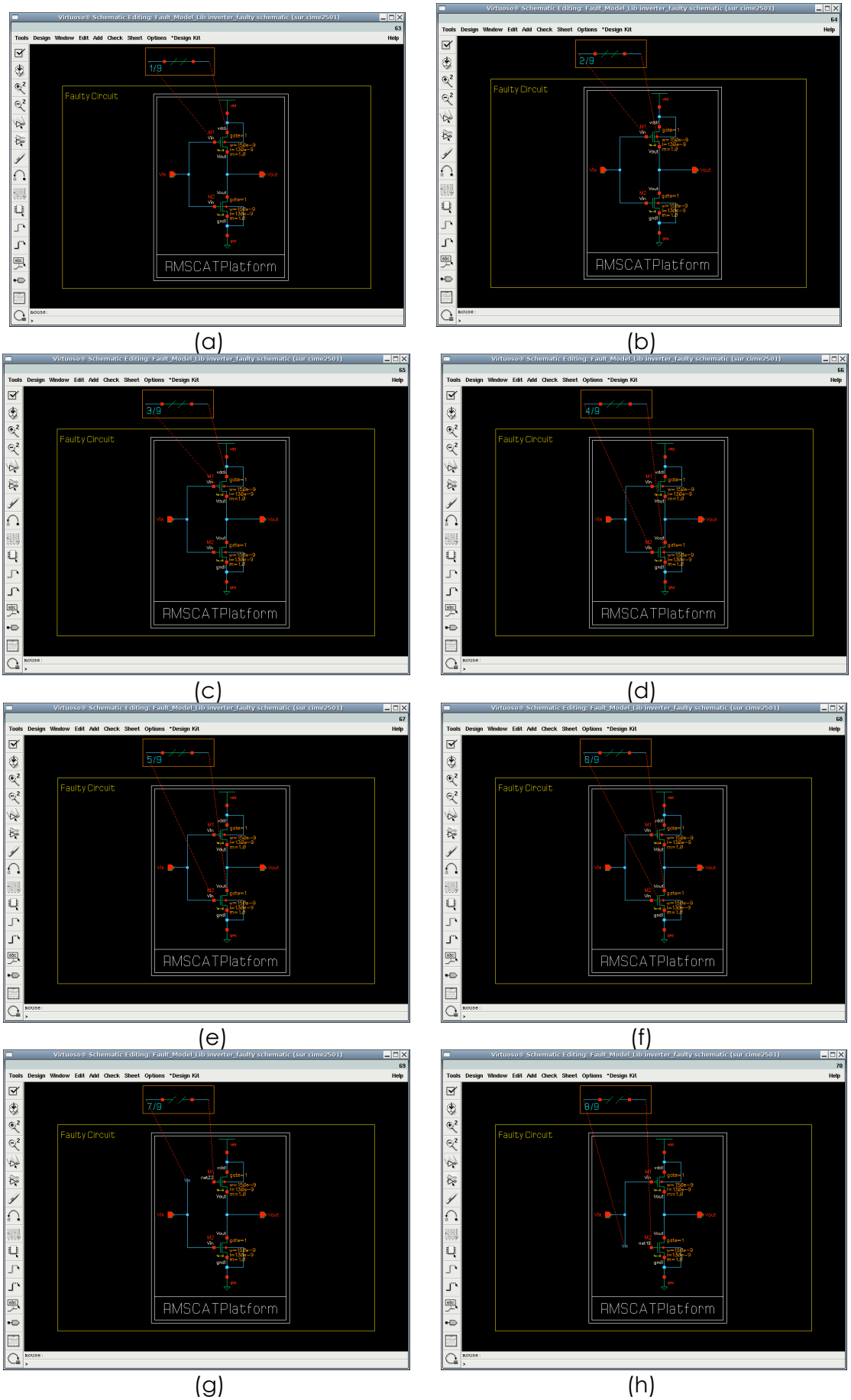



**Figure 3.28**

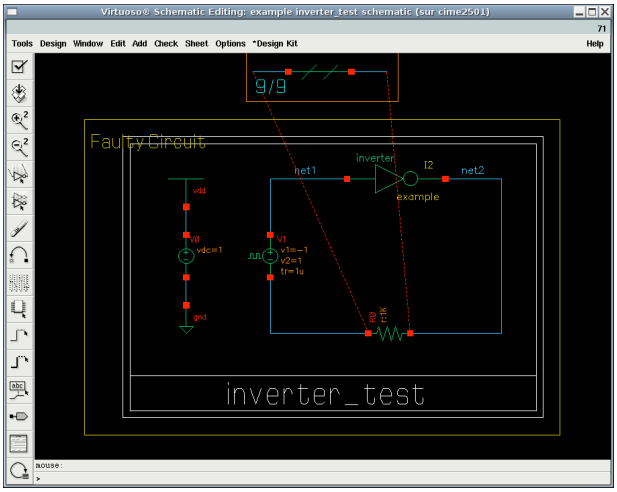
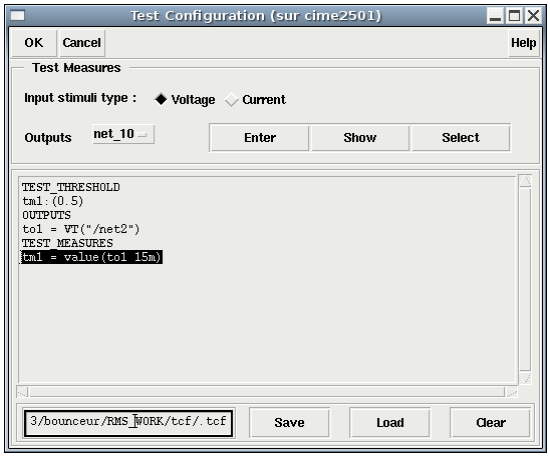


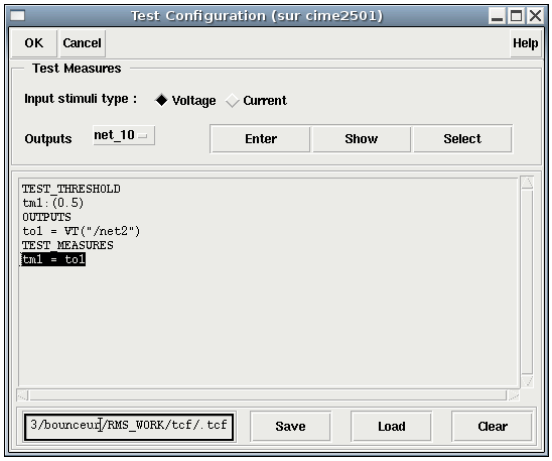

**Figure 3.29**

## 3.6. *Traitement des résultats*

Les résultats peuvent être traités soit en observant les sorties ou bien des valeurs de ces sortie obtenues pour chaque fautes injectée. Il est aussi possible de calculer la couverture de fautes et d'évaluer une technique de test en calculant les différentes métriques de test analogiques.



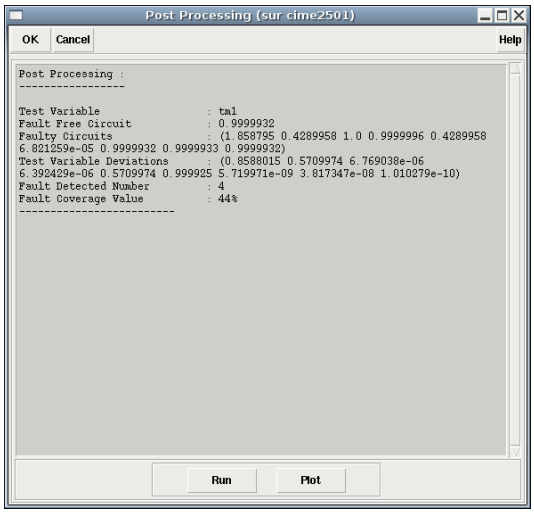

(a)



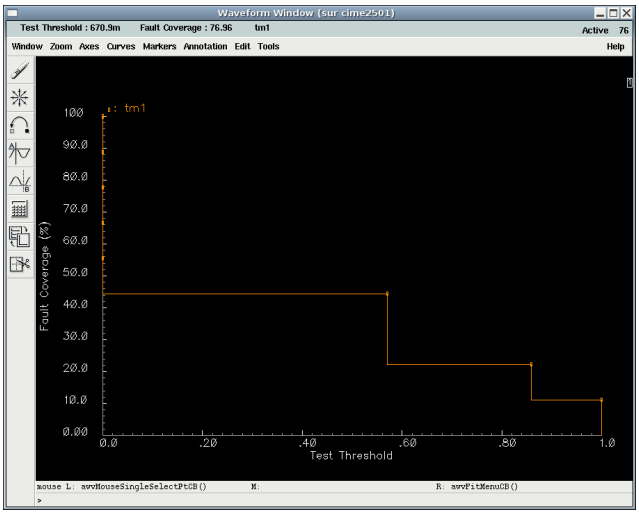

(b)

**Figure 3.30**

La Figure 38(a) montre le programme permettant de calculer et de tracer une couverture de fautes (Figure 39). La Figure 38(b) montre comment modifier ce programme pour observer les différentes sorties (dans ce cas, une seule sortie est considérée) correspondantes à chacune des fautes injectées (Figure 40).


Post Processing (sur cime2501)
OK
Cancel
Help
Post Processing :
Test Variable : tm1
Fault Free Circuit : 0.9999932
Faulty Circuits : (1.858795 0.4289958 1.0 0.9999996 0.4289958 6.821259e-05 0.9999932 0.9999933 0.9999932)
Test Variable Deviations : (0.8588015 0.5709974 6.769038e-06 6.392429e-06 0.5709974 0.999925 5.719971e-09 3.817347e-08 1.010279e-10)
Fault Detected Number : 4
Fault Coverage Value : 44%
Run
Plot

(a)




(b)

**Figure 3.31**

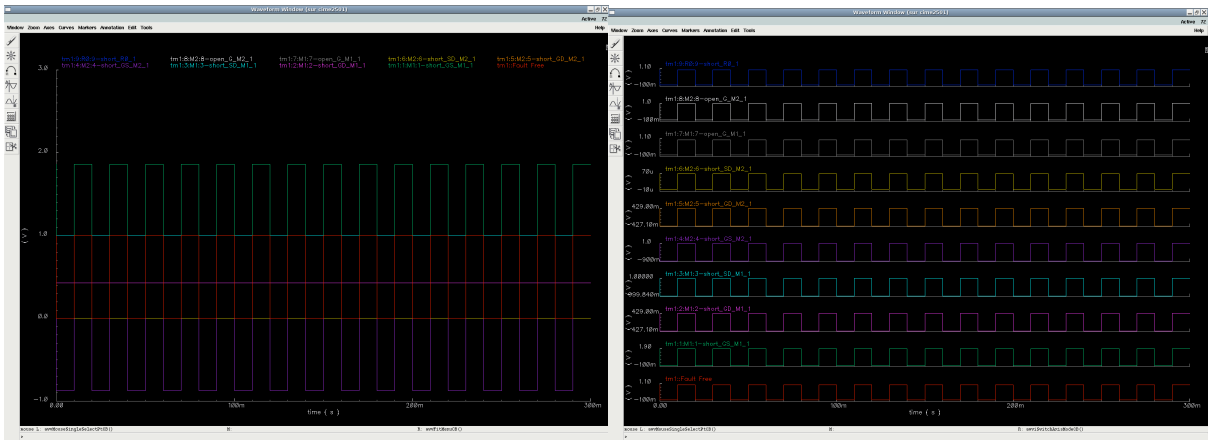


**Figure 3.32**

## 3.7. La simulation de fautes paramétriques

La Figure 3.33 montre un exemple d'une injection de faute paramétrique explicite. Les résultats d'une simulation de fautes paramétriques explicites se traitent comme dans le cas de la simulation de fautes catastrophiques. Pour ce qui est du cas des simulations de fautes paramétriques implicites, ceci sera traité en détail dans les prochaines sections.

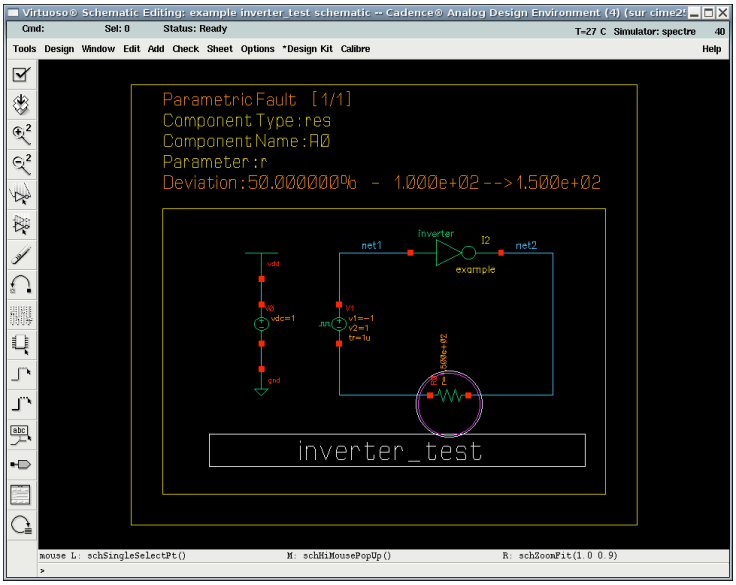



**Figure 3.33**

# Génération des vecteurs de test

## 4. Le circuit : filtre biquadratique différentiel

Dans cette section nous allons montrer comment générer l'ensemble des fréquences minimal d'un vecteur de test. La méthode utilisée se fait uniquement dans le domaine fréquentiel à base de fautes catastrophiques (test structurel). Pour cela, nous allons utiliser un circuit représentant le filtre biquadratique différentiel de la Figure 4.1.

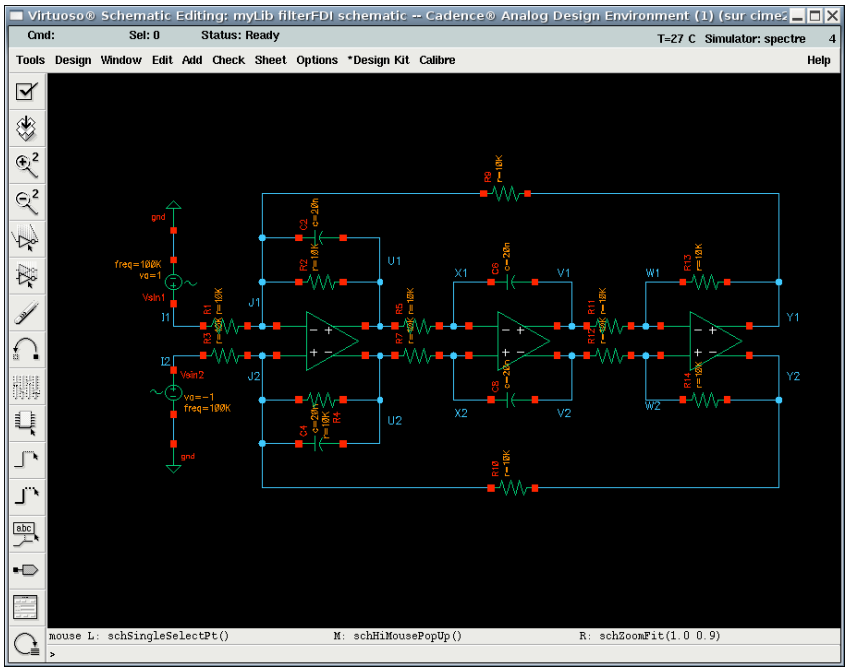

Virtuoso® Schematic Editing: myLib filterFDI schematic -- Cadence® Analog Design Environment (1) (sur cimez
Cmd:
Sel: 0
Status: Ready
T=27 C Simulator: spectre
Tools Design Window Edit Add Check Sheet Options *Design Kit Calibre
Help
mouse L: schSingleSelectPt()
M: schHiMousePopUp()
R: schZoomFit(1.0 0.9)

**Figure 4.1**

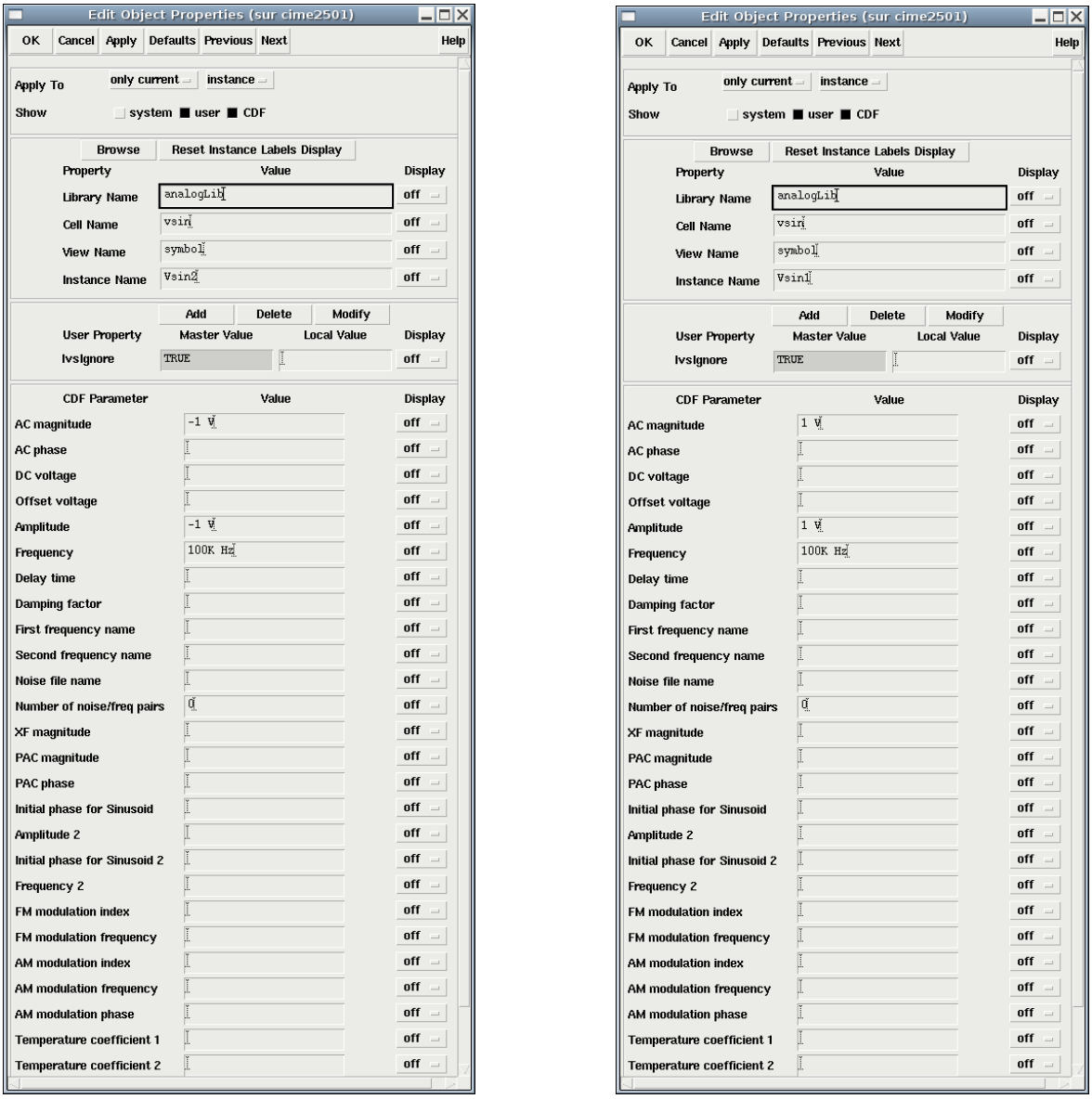
Les configurations des deux vsin (Vsin1 et Vsin2) sont données par les Figure 4.3(a) et (b), respectivement. Les trois amplificateurs opérationnels sont des blocs veriloga. Les blocs sont sous la forme suivante (Figure 4.2):

veriloga.va

File Edit Search Preferences Shell Macro Windows Help

```
// VerilogA for veriloga, fully_diff_amp, veriloga
// Dimitris Mitrovgenis - Ahcene Bounceur
// 02/03/2005

`include "constants.vams"
`include "disciplines.vams"

module model_full_diff(inp,inn,outp,outn);
input inp,inn;
output outp,outn;
electrical inp,inn,outp,outn;

parameter real K=100000;

analog
begin

   V(outn)<+ -(V(inp)-V(inn))*K;
   V(outp)<+ (V(inp)-V(inn))*K;

end
endmodule
```

**Figure 4.2**


Edit Object Properties (sur cime2501)
OK Cancel Apply Defaults Previous Next Help
Apply To only current instance
Show system user CDF
Browse Reset Instance Labels Display
Property Value Display
Library Name analogLib off
Cell Name vsin off
View Name symbol off
Instance Name Vsin2 off
Add Delete Modify
User Property Master Value Local Value Display
IvsIgnore TRUE off
CDF Parameter Value Display
AC magnitude -1 V off
AC phase off
DC voltage off
Offset voltage off
Amplitude -1 V off
Frequency 100K Hz off
Delay time off
Damping factor off
First frequency name off
Second frequency name off
Noise file name off
Number of noise/freq pairs 0 off
XF magnitude off
PAC magnitude off
PAC phase off
Initial phase for Sinusoid off
Amplitude 2 off
Initial phase for Sinusoid 2 off
Frequency 2 off
FM modulation index off
FM modulation frequency off
AM modulation index off
AM modulation frequency off
AM modulation phase off
Temperature coefficient 1 off
Temperature coefficient 2 off
Edit Object Properties (sur cime2501)
OK Cancel Apply Defaults Previous Next Help
Apply To only current instance
Show system user CDF
Browse Reset Instance Labels Display
Property Value Display
Library Name analogLib off
Cell Name vsin off
View Name symbol off
Instance Name Vsin1 off
Add Delete Modify
User Property Master Value Local Value Display
IvsIgnore TRUE off
CDF Parameter Value Display
AC magnitude 1 V off
AC phase off
DC voltage off
Offset voltage off
Amplitude 1 V off
Frequency 100K Hz off
Delay time off
Damping factor off
First frequency name off
Second frequency name off
Noise file name off
Number of noise/freq pairs 0 off
XF magnitude off
PAC magnitude off
PAC phase off
Initial phase for Sinusoid off
Amplitude 2 off
Initial phase for Sinusoid 2 off
Frequency 2 off
FM modulation index off
FM modulation frequency off
AM modulation index off
AM modulation frequency off
AM modulation phase off
Temperature coefficient 1 off
Temperature coefficient 2 off

(a) (b)

**Figure 4.3**

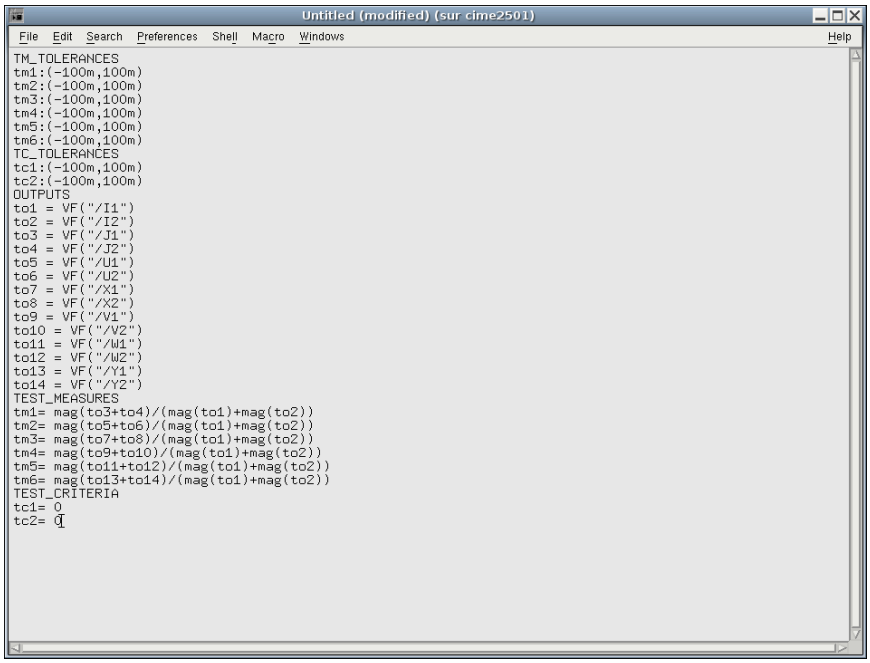
Pour exécuter l'algorithme de génération de vecteurs de test, il faut d'abord effectuer une simulation de fautes catastrophiques. Pour cela, il faut ouvrir l'outil OPTEVAL à partir duquel il faut choisir le filtre présenté ci-dessus. Le programme de test est présenté dans la section suivante.

# 5. Programme de test et simulation de fautes

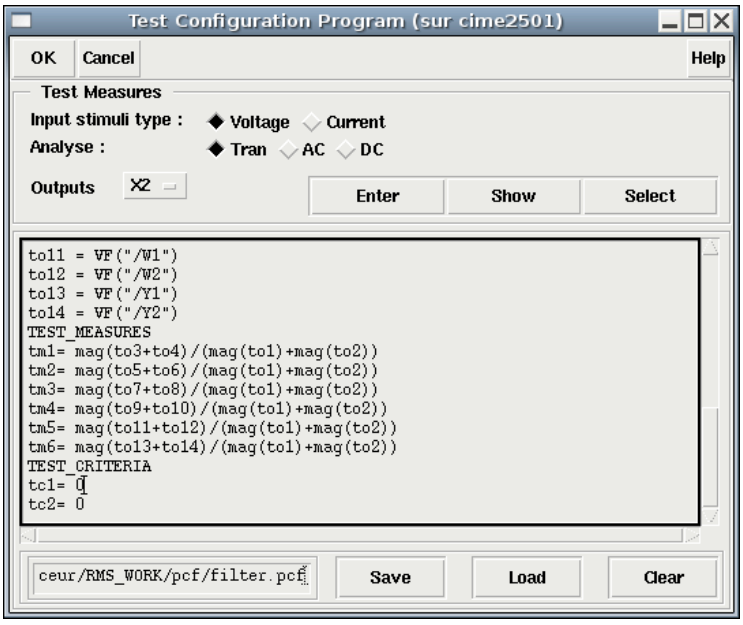
Le programme de test est donné par la Figure 5.1(a). La Figure 5.1(b) montre la fenêtre de l'outil permettant d'entrer ce programme. Ensuite il faut créer une liste de 16 fautes à injecter en créant les fichiers FIDs décrivant comment ces fautes seront injectées. Ces FIDs sont classées et nommés comme suit (Figure 5.2) :

1. Court circuit pour la résistance R1
2. Court circuit pour la résistance R2
3. Court circuit pour la capacité C2
4. Court circuit pour la résistance R9
5. Circuit ouvert pour la résistance R1

6. Circuit ouvert pour la résistance R2
7. Circuit ouvert pour la résistance R9
8. Circuit ouvert pour la capacité C2
9. Court circuit pour la résistance R5
10. Court circuit pour la capacité C6
11. Circuit ouvert pour la résistance R5
12. Circuit ouvert pour la capacité C6
13. Court circuit pour la résistance R11
14. Court circuit pour la résistance R13
15. Circuit ouvert pour la résistance R11
16. Circuit ouvert pour la résistance R13




(a)




(b)

**Figure 5.1**

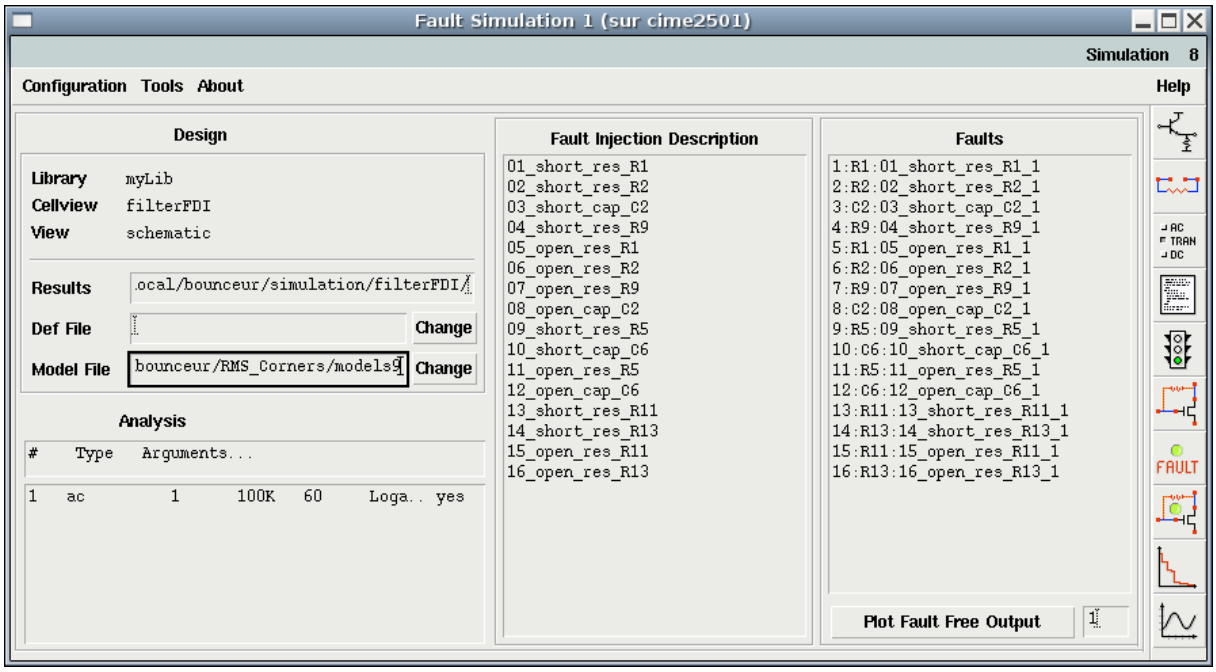
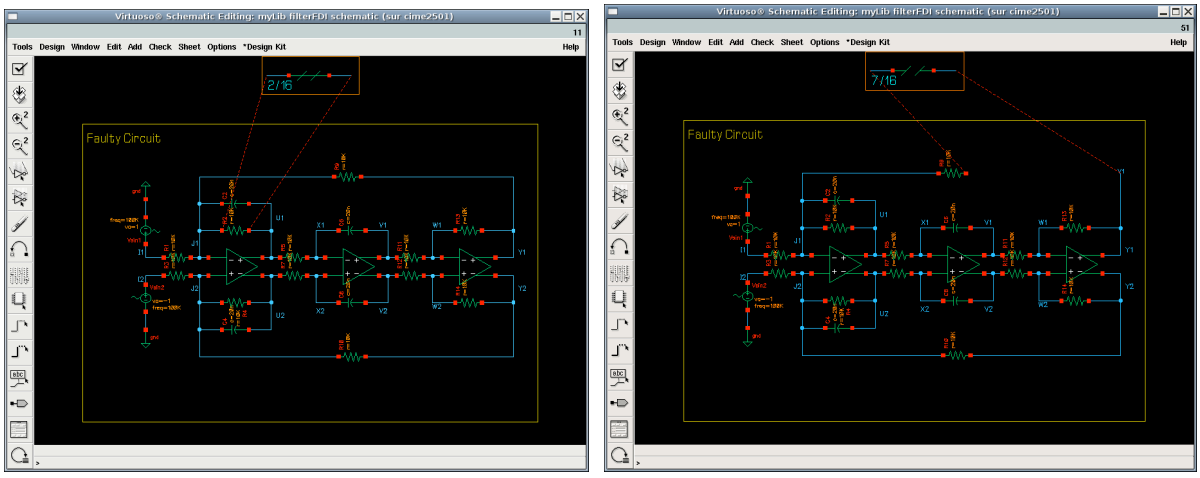



**Figure 5.2**

Il suffit maintenant de lancer la simulation de fautes en lançant en premier la simulation sans fautes puis la simulation avec fautes. Les Figure 5.3(a) et (b) montre l'injection numéro 2 et numéro 7, respectivement.




(a) (b)

**Figure 5.3**

# 6. La génération de vecteurs de test

Une fois la simulation de fautes est terminée, il est donc possible d'exécuter l'algorithme de génération de vecteurs de test.

Avant l'exécution de l'algorithme de génération de vecteurs de test, il faut d'abord choisir le type d'algorithme d'optimisation à utiliser. Cet algorithme permet de résoudre les problèmes de recouvrement nécessaires pour résoudre certains problèmes d'optimisation. Les algorithmes implémentés sont, les algorithmes génétiques et l'algorithme de Balas. Ce dernier permet de résoudre le problème de recouvrement d'une manière exacte. Par contre il peut être utilisé que pour les

problèmes de petite complexité. Pour les gros problèmes, il est recommandé d'utiliser les algorithmes génétiques. Pour choisir ces algorithmes, il faut utiliser le menu configuration de l'outil OPTEVAL puis Optimization Algorithm (Figure 6.1).

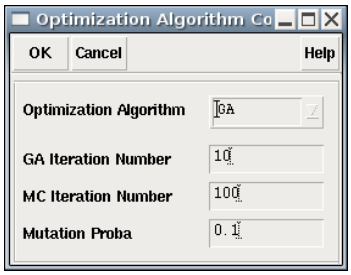



**Figure 6.1**

Maintenant il reste que de cliquer sur le bouton Min Detection Frequencies de l'outil OPTEVAL (Figure 6.2) en choisissant dans la partir Frequencies la plage des fréquences souhaitée du vecteur de test. Et le résultat sera obtenu dans le fichier tpgDetReport.rep. Le nom de ce fichier peut être changé en utilisant la fenêtre Path Configureation qui se trouve dans le menu Configuration de l'outil OPTEVAL (Figure 6.3).

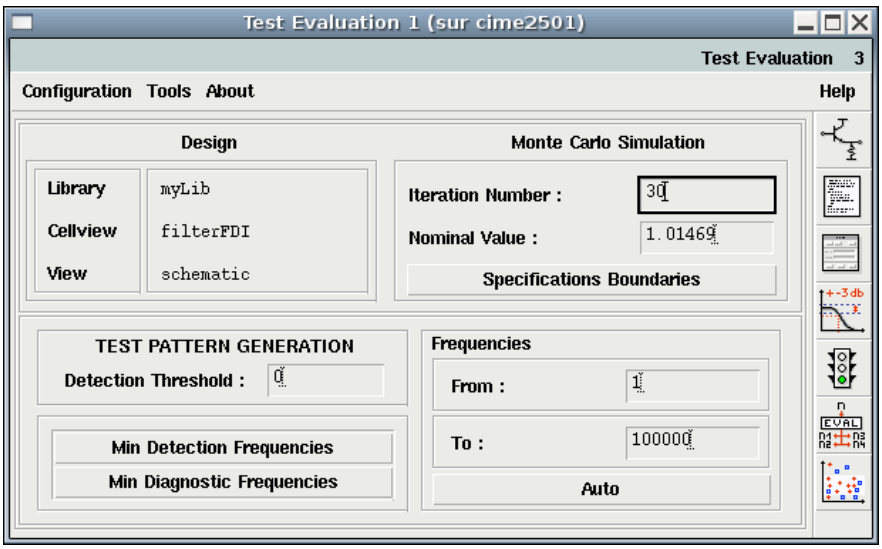



**Figure 6.2**

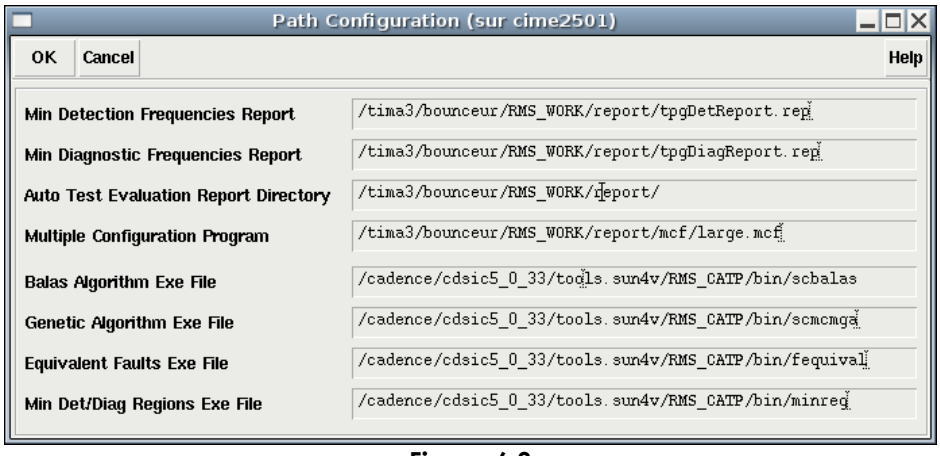



**Figure 6.3**

# Évaluation du test

La plateforme permet d'évaluer une technique de test qu'elle soit structurelle ou bien fonctionnelle. L'évaluation d'un test structurel vient toujours après l'évaluation d'un test fonctionnel dans le but de valider le premier test.

## 6.1. *Évaluation d'un test fonctionnel*

Pour évaluer un test fonctionnel, il est nécessaire de connaître l'ensemble des performances et des spécifications du circuit sous test. Dans le cas où les spécifications ne sont pas connues, il est possible de les calculer à base des simulations Monte Carlo Cadence (en utilisant l'outil de la Figure 6.4 qui se trouve dans le menu Tools→Monte Carlo de Cadence). La plateforme permet une fois les simulations Monte Carlo ont été effectuées de récupérer toutes les valeurs des performances pour chaque itération.

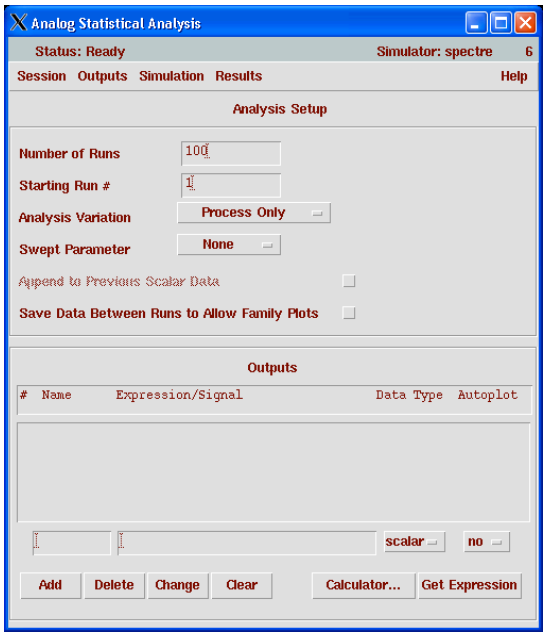



**Figure 6.4**

Une fois les simulations Monte Carlo ont été effectuées, il reste d'utiliser l'outil OPTEVAL pour récupérer les valeurs des performances de chaque performance et de chaque itération Monte Carlo. Pour ce faire, il faut ouvrir la fenêtre Test Evaluation Results (Figure 6.5) qui se trouve dans le menu Tools. Une fois cette fenêtre ouverte, il faut cocher la case Monte Carlo, puis Save, puise Performances. Il n'est pas vraiment nécessaire dans ce cas de cocher la case Test Criteria. Cliquez ensuite sur le bouton Run Test Evaluation. Les résultats obtenus sont sauvegardés par défaut dans le répertoire report. Le chemin de ce répertoire peut être changé en utilisant la fenêtre de la Figure 6.3. Les fichiers créés sont :
specValues.rep : ce fichier contient la valeur de chaque performance et de chaque itération Monte Carlo.
specVerif.rep : ce fichier contient les valeurs vrai ou faux qui détermine pour chaque itération si une performance est vérifiée ou non.

Une fois ses valeurs sont connues, à l'aide d'un outil ou d'un logiciel statistique, il est possible de connaître la loi de chaque performance et ainsi de déterminer les intervalles de tolérance, qu'on appellera spécifications, pour chacune d'elle.

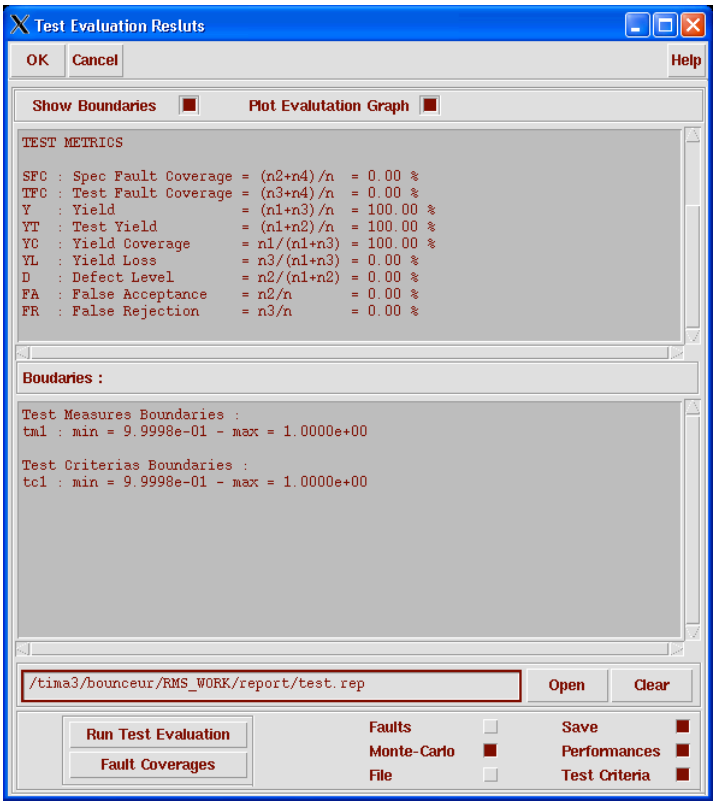

Test Evaluation Resluts
OK
Cancel
Help
Show Boundaries
Plot Evalutation Graph
TEST METRICS
SFC : Spec Fault Coverage = (n2+n4)/n = 0.00 %
TFC : Test Fault Coverage = (n3+n4)/n = 0.00 %
Y : Yield = (n1+n3)/n = 100.00 %
YT : Test Yield = (n1+n2)/n = 100.00 %
YC : Yield Coverage = n1/(n1+n3) = 100.00 %
YL : Yield Loss = n3/(n1+n3) = 0.00 %
D : Defect Level = n2/(n1+n2) = 0.00 %
FA : False Acceptance = n2/n = 0.00 %
FR : False Rejection = n3/n = 0.00 %
Boudaries :
Test Measures Boundaries :
tm1 : min = 9.9998e-01 - max = 1.0000e+00
Test Criterias Boundaries :
tc1 : min = 9.9998e-01 - max = 1.0000e+00
/tima3/bounceur/RMS_WORK/report/test.rep
Open
Clear
Run Test Evaluation
Fault Coverages
Faults
Monte-Carlo
File
Save
Performances
Test Criteria

**Figure 6.5**

## 6.2. *Évaluation d'un test structurel*

Cette étape ressemble à la première, sauf qu'au lieu de faire une simulation de type Monte Carlo, il faut faire une simulation de fautes. Ensuite, dans la fenêtre de l'évaluation de test (Figure 6.5) il faut cocher la case Faults au lieu de Monte Carlo.

# Codification numérique d'un vecteur analogique

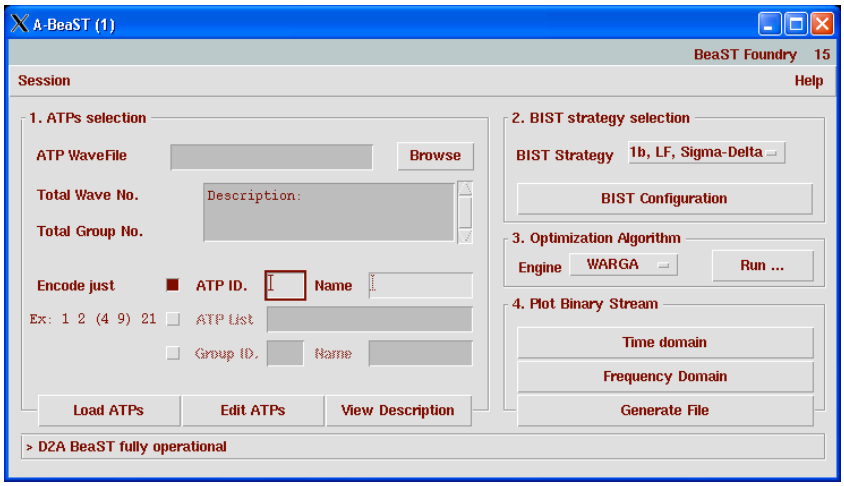



**Figure 6.6**

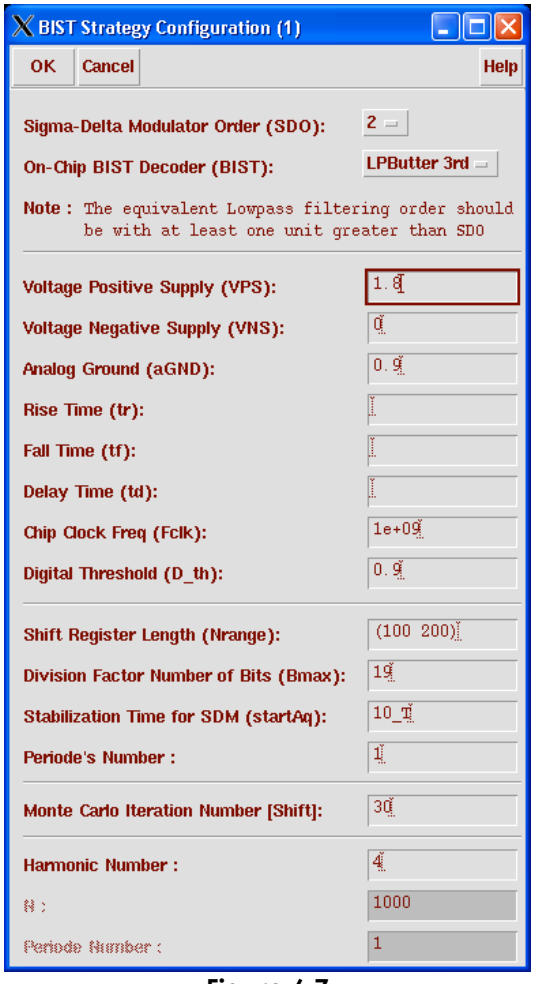



**Figure 6.7**

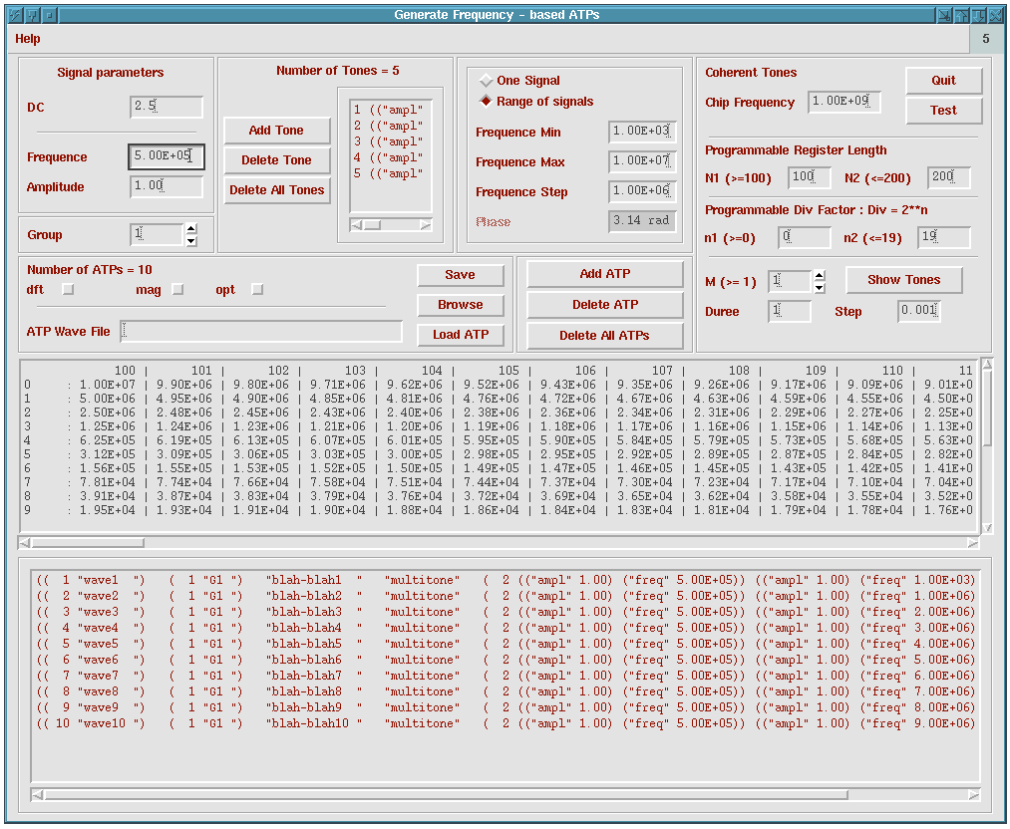



**Figure 6.8**

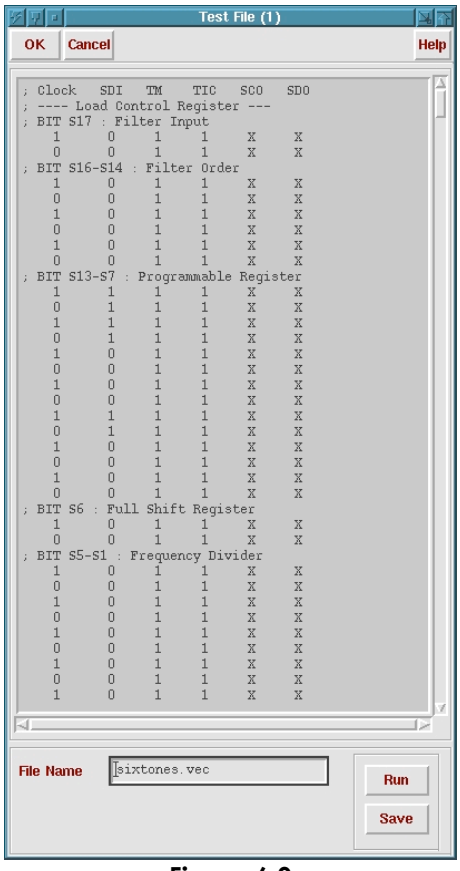

Test File (1)
OK
Cancel
Help
; Clock SDI TM TIC SCO SDO
; ---- Load Control Register ---
; BIT S17 : Filter Input
1 0 1 1 X X
0 0 1 1 X X
; BIT S16-S14 : Filter Order
1 0 1 1 X X
0 0 1 1 X X
1 0 1 1 X X
0 0 1 1 X X
1 0 1 1 X X
0 0 1 1 X X
; BIT S13-S7 : Programmable Register
1 1 1 1 X X
0 1 1 1 X X
1 1 1 1 X X
0 1 1 1 X X
1 0 1 1 X X
0 0 1 1 X X
1 0 1 1 X X
0 0 1 1 X X
1 1 1 1 X X
0 1 1 1 X X
1 0 1 1 X X
0 0 1 1 X X
1 0 1 1 X X
0 0 1 1 X X
; BIT S6 : Full Shift Register
1 0 1 1 X X
0 0 1 1 X X
; BIT S5-S1 : Frequency Divider
1 0 1 1 X X
0 0 1 1 X X
1 0 1 1 X X
0 0 1 1 X X
1 0 1 1 X X
0 0 1 1 X X
1 0 1 1 X X
0 0 1 1 X X
1 0 1 1 X X
File Name
sixtones.vec
Run
Save

**Figure 6.9**

# Calcul des noyaux de Volterra

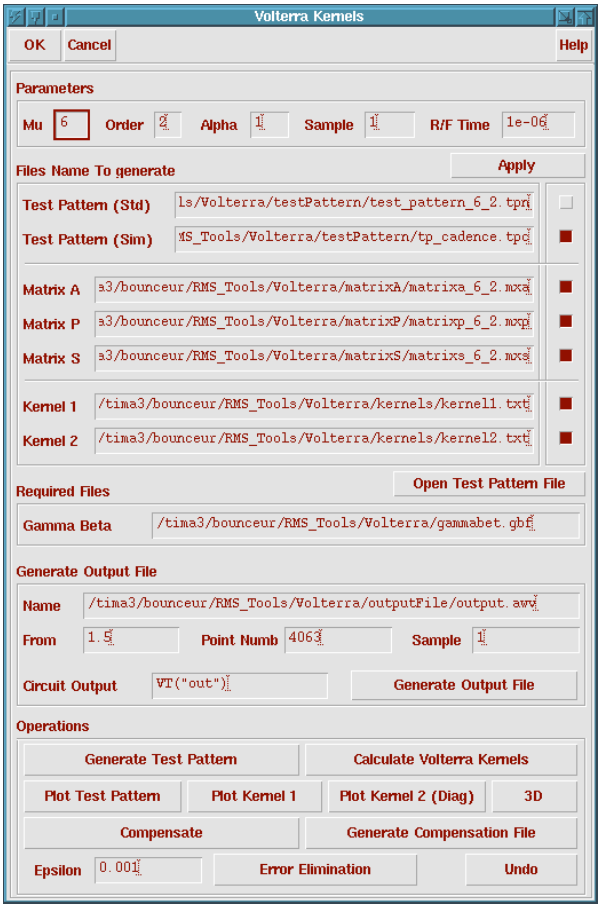



**Figure 6.10**

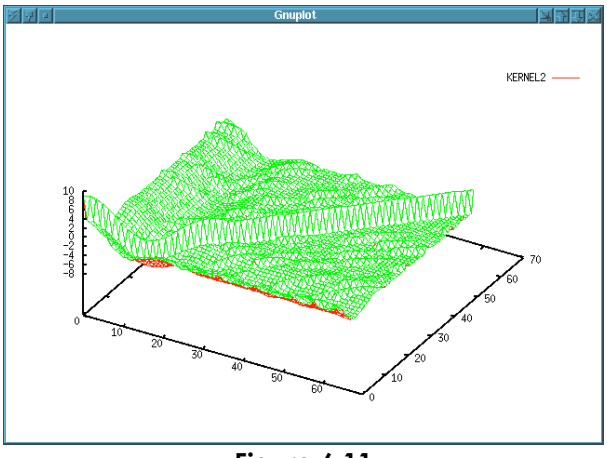



**Figure 6.11**